



ユーザーガイド

Amazon ElastiCache for Redis



API バージョン 2015-02-02

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon ElastiCache for Redis: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

ElastiCache for Redis とは	1
サーバーレスキャッシュ	1
独自設計型クラスター	2
関連サービス	2
仕組み	3
キャッシュとキャッシュエンジン	3
デプロイの選択	8
機能の比較	10
ElastiCache のリソース	15
AWS リージョンとアベイラビリティゾーン	17
ユースケース	18
インメモリデータストア	19
ゲームのリーダーボード (Redis ソートセット)	20
メッセージング (Redis パブリッシュ/サブスクライブ)	22
推奨データ (Redis ハッシュ)	25
その他の Redis の用途	25
ElastiCache のお客様の声	25
Redis ElastiCache 用スタートガイド	26
セットアップ	26
にサインアップする AWS アカウント	26
管理アクセスを持つユーザーを作成する	27
プログラマ的なアクセス権を付与する	28
アクセス許可の設定	30
EC2 をセットアップする	31
ネットワークアクセスを許可する	31
redis-cli をセットアップする	32
キャッシュを作成します。	33
データの読み取りと書き込み	34
クリーンアップ	36
次のステップ	37
ElastiCache および AWS SDK の使用開始	37
Python と ElastiCache	38
チュートリアル:Amazon VPC ElastiCache 内のアマゾンにアクセスするための Lambda 関数の設定	56

ステップ 1: サーバーレスキャッシュを作成する	56
ステップ 2: Lambda 関数を作成する	59
ステップ 3: Lambda 関数をテストする	63
ステップ 4: クリーンアップ (オプション)	64
独自の ElastiCache クラスターの設計	66
コンポーネントと機能	66
ノード	67
ElastiCache Redis シャード用	67
ElastiCache Redis クラスター用	68
ElastiCache Redis レプリケーション用	70
AWS リージョンとアベイラビリティゾーン	72
ElastiCache Redis エンドポイント用	72
パラメータグループ	73
ElastiCache Redis のセキュリティ用	74
[サブネットグループ]	74
ElastiCache Redis バックアップ用	74
イベント	75
ElastiCache for Redis の用語	76
独自のクラスターの設計	79
セットアップ	79
ステップ 1: サブネットグループの作成	79
ステップ 2: クラスターを作成する	82
ステップ 3: クラスターへのアクセスの許可	89
ステップ 4: クラスターのノードに接続する	92
ステップ 5: クラスターを削除する	100
チュートリアルと動画	101
次のステップ	107
ノードの管理	108
ElastiCache ノードステータスの表示	109
Redis のノードとシャード	113
ノードに接続する	116
サポートされているノードの種類	119
ノードの再起動 (クラスターモードが無効な場合のみ)	136
ノードの置換	138
リザーブドノード	145
以前の世代のノードの移行	156

クラスタの管理	159
ネットワークタイプの選択	161
データ階層化	165
クラスタを準備する	172
クラスタの作成	179
クラスタの詳細を表示する	190
クラスタの変更	202
クラスタへのノードの追加	207
クラスタからのノードの削除	215
保留中のノードの追加または削除オペレーションのキャンセル	223
クラスタの削除	224
クラスタまたはレプリケーショングループへのアクセス	227
接続エンドポイントの検索	233
シャード	245
Memcached キャッシュと Redis の独自設計型キャッシュの比較	250
ElastiCache へのオンライン移行	256
概要	256
移行手順	257
ソースおよびターゲット Redis ノードの移行の準備	257
データ移行のテスト	259
移行の開始	259
データ移行の進行状況の確認	261
データ移行の完了	261
コンソールを使用したオンラインデータ移行の実行	262
リージョンとアベイラビリティーゾーンの選択	264
ノードの配置	266
サポートされているリージョンおよびエンドポイント	266
Local Zones の使用	271
Outposts の使用	273
の使用 ElastiCache	277
スナップショットおよび復元	277
制約	278
独自設計型クラスタのバックアップがパフォーマンスに与える影響	279
自動バックアップのスケジュール	280
手動バックアップの取得	281
最終バックアップの作成	287

バックアップの詳細の表示	290
バックアップのコピー	292
バックアップのエクスポート	295
バックアップからの復元	303
バックアップの削除	305
バックアップへのタグ付け	307
バックアップによる独自設計型クラスターのシード	308
エンジンバージョンとアップグレード	317
エンジンバージョンとアップグレード	318
サポートされる Redis のバージョン	323
Redis バージョンのサポート終了スケジュール	337
エンジンバージョンのアップグレード方法	321
ブロックされたエンジンのアップグレードの解決	321
メジャーバージョンの動作と互換性の違い	340
ベストプラクティスとキャッシュ戦略	344
Redis の使用	344
Redis クライアントに関するベストプラクティス	384
予約メモリの管理	411
独自設計型クラスターを使用する場合のベストプラクティス	417
Redis のベストプラクティス	423
キャッシュ戦略	425
独自設計型クラスターの管理	430
Redis クラスター Auto ElastiCache Scaling	431
クラスターモードの変更	478
グローバルデータストアを使用した AWS リージョン間のレプリケーション	481
レプリケーショングループを使用する高可用性	509
メンテナンスの管理	596
パラメータグループを使用したエンジンパラメータの設定	599
Redis ElastiCache のスケーリング	699
ElastiCache サーバーレスのスケーリング	699
スケーリング制限を設定してコストを管理する	700
ElastiCache Serverless による事前スケーリング	700
コンソールとを使用したスケーリング制限の設定 AWS CLI	701
Redis ElastiCache の自己設計クラスターのスケーリング	702
ElastiCache for Redis での JSON の使用開始	774
Redis JSON データ型の概要	775

JSON コマンド	787
ElastiCache リソースのタグ付け	828
タグによるコストのモニタリング	840
AWS CLI を使用したタグの管理	841
ElastiCache API を使用したタグの管理	845
Amazon ElastiCache Well-Architected レンズ	847
オペレーショナルエクセレンスの柱	848
セキュリティの柱	856
信頼性の柱	863
パフォーマンス効率の柱	869
コスト最適化の柱	879
トラブルシューティング	886
接続の問題	886
Redis クライアントエラー	887
ElastiCache Serverless での高レイテンシーのトラブルシューティング	887
ElastiCache Serverless でのスロットリングの問題のトラブルシューティング	889
関連トピック	890
その他のトラブルシューティング手順	890
セキュリティグループ	891
ネットワーク ACL	891
ルートテーブル	893
DNS 解決	893
サーバー側の診断に関する問題の特定	894
ネットワーク接続性の検証	899
ネットワーク関連の制限	901
CPU 使用率	903
サーバー側からの接続が終了している	906
Amazon EC2 インスタンスのクライアント側のトラブルシューティング	907
1 つのリクエストを完了するのにかかった時間の解説	908
セキュリティ	912
データ保護	913
Amazon ElastiCache のデータセキュリティ	913
インターネットトラフィックのプライバシー	986
Amazon VPC と ElastiCache のセキュリティ	986
Amazon ElastiCache API とインターフェイス VPC エンドポイント (AWS PrivateLink) ...	1011
サブネットおよびサブネットグループ	1014

Identity and Access Management	1023
対象者	1023
アイデンティティを使用した認証	1024
ポリシーを使用したアクセスの管理	1027
Amazon と IAM ElastiCache の連携方法	1030
アイデンティティベースポリシーの例	1037
トラブルシューティング	1040
アクセスコントロール	1042
アクセス管理の概要	1043
コンプライアンス検証	1085
詳細情報	1087
耐障害性	1087
障害の軽減	1088
インフラストラクチャセキュリティ	1091
サービスの更新	1091
サービスアップデートの管理	1092
セキュリティ上の脆弱性への対処	1097
ロギングとモニタリング	1099
サーバーレスのメトリクスとイベント	1099
サーバーレスメトリクス	1099
サーバーレスイベント	1108
独自設計型のクラスターメトリクスとイベント	1123
独自設計型クラスターのメトリクス	1123
独自設計型クラスターのイベント	1123
ログ配信	1132
使用状況のモニタリング	1145
Amazon SNS イベントモニタリング	1174
AWS CloudTrail を使用した Amazon ElastiCache API コール	1191
CloudTrail の Amazon ElastiCache 情報	1191
Amazon ElastiCache ログファイルエントリの理解	1192
クォータ	1197
リファレンス	1199
ElastiCache API の使用	1199
クエリ API を使用する	1199
利用可能なライブラリ	1203
アプリケーションのトラブルシューティング	1203

AWS CLIElastiCache をセットアップする	1204
前提条件	1205
コマンドラインツールを入手する	1206
ツールを設定する	1207
ツールでの認証情報の指定	1208
環境変数	1209
エラーメッセージ	1210
通知	1211
一般的な ElastiCache の通知	1212
ElastiCache for Redis 固有の通知	1212
ElastiCache for Redis ドキュメント履歴	1213
AWS 用語集	1246
.....	mccxlvii

Amazon ElastiCache for Redis とは

Amazon ElastiCache for Redis ユーザーガイド へようこそ。Amazon ElastiCache は、クラウド内の分散インメモリデータストアまたはキャッシュ環境を簡単にセットアップ、管理、スケーリングできるウェブサービスです。高性能かつスケーラブルで費用対効果の高いキャッシュソリューションを提供します。同時に、分散キャッシュ環境のデプロイと管理に関連する複雑さを排除するのに役立ちます。

Amazon は ElastiCache 2 つの形式で運用できます。サーバーレスキャッシュで始めるか、独自のキャッシュクラスターを設計するかを選択できます。

Note

Amazon は、Redis エンジンと Memcached エンジンの両方で ElastiCache 動作します。関心のあるエンジンのガイドを使用してください。必要なエンジンがわからない場合は、このガイドの「[Memcached キャッシュと Redis の独自設計型キャッシュの比較](#)」を参照してください。

サーバーレスキャッシュ

ElastiCache for Redis にはサーバーレスキャッシュが用意されているため、アプリケーションの Redis ベースのキャッシュの追加と運用が簡単になります。ElastiCache Redis Serverless を使用すると、可用性の高いキャッシュを 1 分以内に作成できるため、インスタンスのプロビジョニングやノードやクラスターの設定が不要になります。デベロッパーは、ElastiCache コンソール、SDK、または CLI を使用してキャッシュ名を指定することで、サーバーレスキャッシュを作成できます。

ElastiCache for Redis Serverless では、キャッシュ容量を計画および管理する必要もなくなります。ElastiCache for Redis は、キャッシュのメモリを常にモニタリングします。コンピューティング、アプリケーションで使用されるとネットワーク帯域幅、およびは、アプリケーションのニーズに合わせてスケーリングします。ElastiCache for Redis は、デベロッパー向けにシンプルなエンドポイントエクスペリエンスを提供します。基盤となるキャッシュインフラストラクチャとクラスター設計を抽象化すること。ElastiCache for Redis はハードウェアプロビジョニングを管理します。モニタリング、ノードの交換 およびソフトウェアのパッチ適用は、自動的かつ透過的に行われます。アプリケーション開発に集中できるように、キャッシュを操作するのではなく。

ElastiCache for Redis Serverless は、Redis 7.1 以降と互換性があります。

Redis クラスター ElastiCache 用に独自の を設計する

ElastiCache for Redis クラスターをきめ細かく制御する必要がある場合は、独自の Redis クラスターを設計することを選択できます ElastiCache。ElastiCache では、クラスターのア AWS ベイラビリティゾーン全体でノードタイプ、ノード数、ノード配置を選択することで、クラスターを設計できます。ElastiCache はフルマネージドサービスであるため、クラスターのハードウェアプロビジョニング、モニタリング、ノード交換、およびソフトウェアパッチ適用を自動的に管理します。

Redis クラスター ElastiCache 用に独自の を設計すると、クラスターの柔軟性と制御が向上します。例えば、クラスターを必要に応じてシングル AZ 可用性で運用するか、マルチ AZ 可用性で運用するかを選択できます。Redis を水平スケーリングを有効にするクラスターモードで実行するか、クラスターモードなしで垂直スケーリングのみを有効にするかを選択できます。独自のクラスターを設計するときは、キャッシュにアプリケーションが必要とする十分な容量が確保されるように、ノードの種類と数を正しく選択する必要があります。Redis クラスターに新しいソフトウェアパッチを適用するタイミングも選択できます。

Redis クラスター ElastiCache 用に独自の を設計する場合、Redis 3.0 以降を実行することを選択できます。

関連サービス

[Amazon MemoryDB for Redis](#)

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

- ElastiCache for Redis は、Redis を使用して他のデータベースやデータストアからデータをキャッシュするために一般的に使用されるサービスです。既存のプライマリデータベースまたはデータストアによるデータアクセスを高速化 (マイクロ秒単位の読み取り/書き込みパフォーマンス) させたいキャッシュワークロードには、ElastiCache for Redis を検討する必要があります。また、Redis のデータ構造と API を使用してプライマリデータベースまたはデータストアに保存されているデータにアクセスするユースケースには、ElastiCache for Redis を検討する必要があります。
- Amazon MemoryDB for Redis は、超高速のプライマリデータベースを必要とするワークロード向けの、耐久性に優れたインメモリデータベースです。ワークロードが超高速のパフォーマンス (マイクロ秒の読み取りと 1 桁ミリ秒の書き込みレイテンシー) を提供する耐久性のあるデータベースを必要とする場合は、MemoryDB の使用を検討する必要があります。また、Redis のデータ構造と API を使用して、プライマリで耐久性の高いデータベースを使用してアプリケーションを構築したい場合も、MemoryDB が適している場合があります。最後に、MemoryDB を使用してアプリ

ケーションアーキテクチャを簡素化し、データベースの使用をキャッシュに置き換えて耐久性とパフォーマンスを向上させることで、コストを削減することを検討してください。

「[Amazon RDS](#)」

ElastiCache for Redis は、頻繁にアクセスされるデータをキャッシュに保存することで、データベースのコストを節約します。アプリケーションの読み取りスループット要件が高い場合は、基盤となるデータベースをスケールする代わりに ElastiCache を使用することで、高いスケーラビリティ、高速パフォーマンス、およびデータストレージコストの削減を実現できます。

仕組み

ここでは、ElastiCache for Redis デプロイの主なコンポーネントの概要を確認できます。

キャッシュとキャッシュエンジン

キャッシュは、キャッシュされたデータを保存するために使用できるインメモリデータストアです。通常、アプリケーションは応答時間を最適化するために、頻繁にアクセスされるデータをキャッシュにキャッシュします。ElastiCache for Redis には、サーバーレスクラスターと自己設計クラスターの 2 つのデプロイオプションがあります。「[デプロイオプションの選択](#)」を参照

Note

Amazon は、Redis エンジンと Memcached エンジンの両方で ElastiCache 動作します。関心のあるエンジンのガイドを使用してください。必要なエンジンがわからない場合は、このガイドの「[Memcached キャッシュと Redis の独自設計型キャッシュの比較](#)」を参照してください。

トピック

- [ElastiCache for Redis の仕組み](#)
- [料金ディメンション](#)
- [ElastiCache Redis バックアップ用の](#)

ElastiCache for Redis の仕組み

ElastiCache for Redis サーバーレス

ElastiCache for Redis Serverless では、容量計画、ハードウェア管理、クラスター設計を気にすることなく、キャッシュを作成できます。キャッシュの名前を指定するだけで、Redis クライアントで設定できるエンドポイントが 1 つ用意され、キャッシュへのアクセスを開始できます。

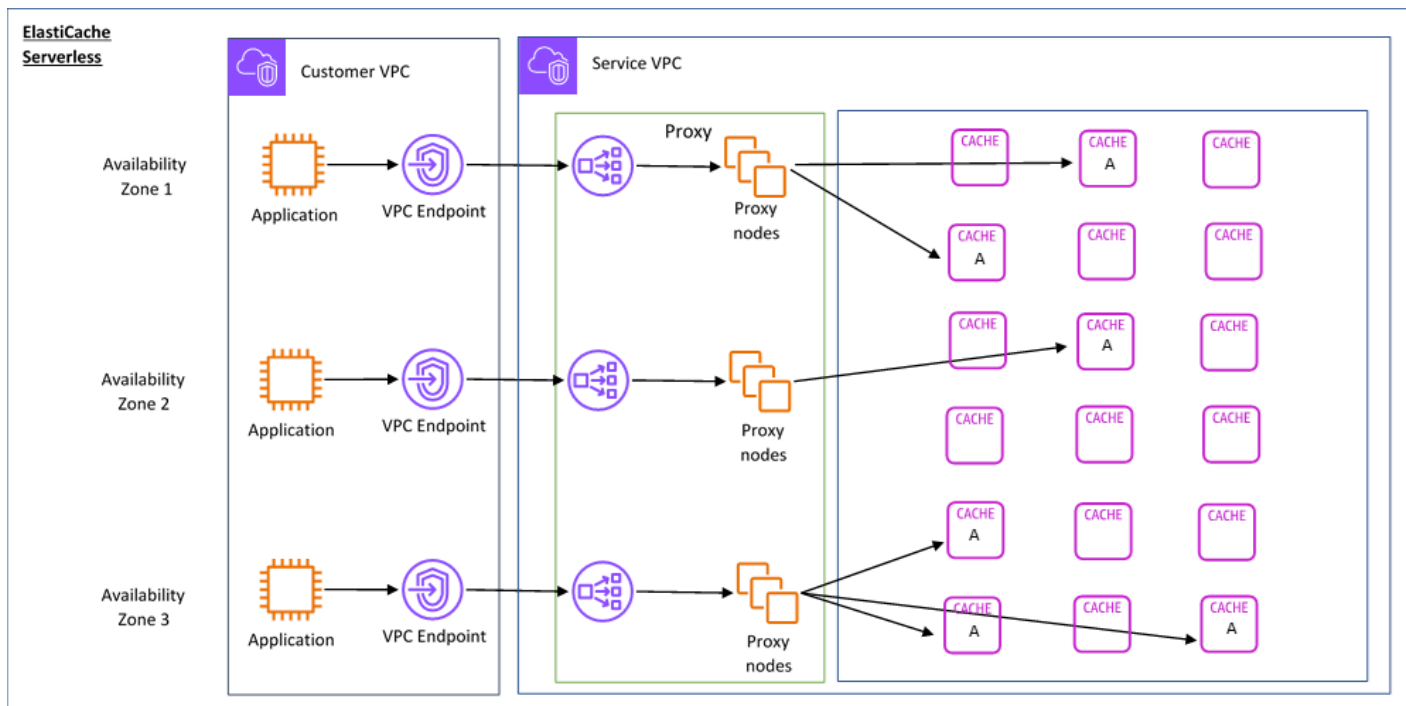
Note

ElastiCache for Redis Serverless は、クラスターモードで Redis を実行し、TLS と Redis クラスタープロトコルの両方をサポートする Redis クライアントとのみ互換性があります。

主な利点

- **容量計画なし**：ElastiCache Serverless を使用すると、容量を計画する必要がなくなります。ElastiCache Serverless は、キャッシュのメモリ、コンピューティング、ネットワーク帯域幅の使用率を継続的にモニタリングし、垂直方向と水平方向の両方にスケールアップします。これにより、キャッシュノードのサイズが大きくなるのと同時に、並行してスケールアウトオペレーションが開始され、キャッシュが常にアプリケーションの要件を満たすように拡張できるようになります。
- **Pay-per-use**：ElastiCache Serverless では、キャッシュ上のワークロードによって保存および使用されるデータに対して料金が発生します。[料金ディメンション](#) を参照してください。
- **高可用性**：ElastiCache サーバーレスは、高可用性を実現するために、複数のアベイラビリティーゾーン (AZ) 間でデータを自動的にレプリケートします。基盤となるキャッシュノードを自動的に監視し、障害が発生した場合はそれらを置き換えます。すべてのキャッシュで、99.99% の可用性 SLA を提供します。
- **ソフトウェアの自動アップグレード**：ElastiCache サーバーレスは、アプリケーションに影響を与えることなく、キャッシュを最新のマイナーおよびパッチソフトウェアバージョンに自動的にアップグレードします。新しい Redis メジャーバージョンが利用可能になると、ElastiCache から通知が送信されます。
- **セキュリティ**：サーバーレスでは、転送中および保管中のすべてのデータが暗号化されます。保管中のデータを暗号化するには、サービス管理キーを使用するか、独自のカスタマー管理キーを使用できます。

次の図は、ElastiCache サーバーレスの仕組みを示しています。



新しいサーバーレスキャッシュを作成すると、は VPC 内の任意のサブネットに Virtual Private Cloud (VPC) エンドポイント ElastiCache を作成します。アプリケーションはこれらの VPC エンドポイントを介してキャッシュに接続できます。

ElastiCache Serverless では、アプリケーションが接続する単一の DNS エンドポイントを受け取ります。エンドポイントへの新しい接続をリクエストすると、ElastiCache Serverless はプロキシレイヤーを介してすべてのキャッシュ接続を処理します。プロキシレイヤーでは、基盤となるクラスターが変更された場合にクライアントがクラスタートポロジを再検出する必要がないため、複雑なクライアント設定が軽減されます。プロキシレイヤーは、Network Load Balancer を使用して接続を処理する一連のプロキシノードです。アプリケーションが新しいキャッシュ接続を作成すると、リクエストは Network Load Balancer によってプロキシノードに送信されます。アプリケーションがキャッシュコマンドを実行すると、アプリケーションに接続されているプロキシノードがキャッシュ内のキャッシュノードでリクエストを実行します。プロキシレイヤーは、キャッシュクラスタのトポロジとノードをクライアントから抽象化します。これにより、ElastiCache は、アプリケーションの可用性に影響を与えたり、接続をリセットしたりすることなく、キャッシュノードのインテリジェントな負荷分散、スケールアウト、新しいキャッシュノードの追加、障害発生時のキャッシュノードの置き換え、キャッシュノード上のソフトウェアの更新を行うことができます。

自己設計 ElastiCache クラスタ

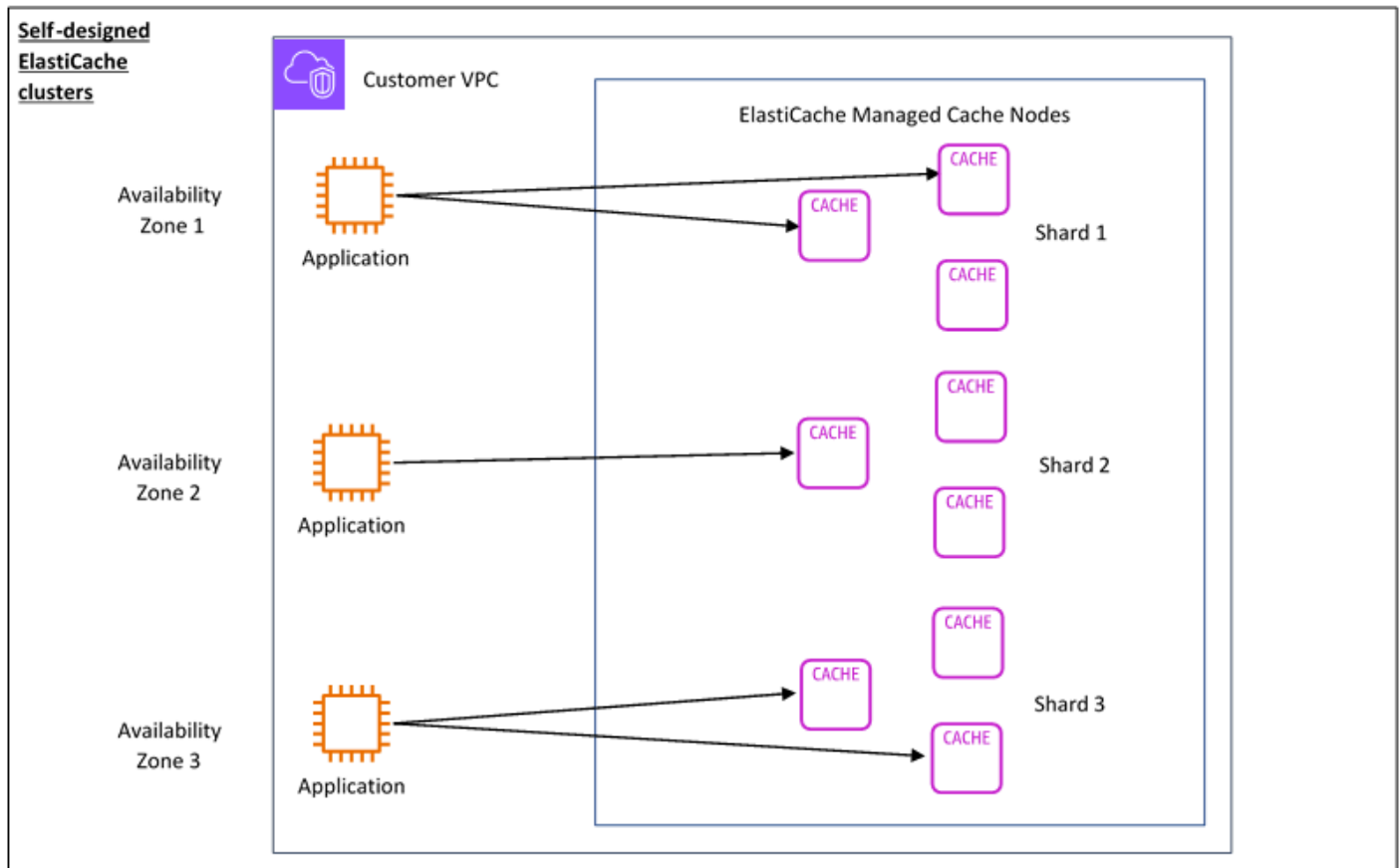
ElastiCache クラスタの キャッシュノードファミリー、サイズ、ノード数を選択することで、独自のクラスタを設計できます。独自のクラスタを設計することで、きめ細かな制御が可能にな

り、キャッシュ内のシャードの数と各シャードのノード (プライマリとレプリカ) の数を選択できます。Redis は、複数のシャードを含むクラスターを作成してクラスターモードで運用するか、1つのシャードで非クラスターモードで運用するかを選択できます。

主な利点

- 独自のクラスターを設計する： を使用すると ElastiCache、独自のクラスターを設計し、キャッシュノードを配置する場所を選択できます。例えば、高可用性と引き換えに低レイテンシーを追求したいアプリケーションがある場合、キャッシュノードを単一の AZ にデプロイできます。あるいは、複数の AZ にまたがるノードを含むクラスターを設計して、高可用性を実現することもできます。
- きめ細かな制御: 独自のクラスターを設計すると、キャッシュの設定をより細かく微調整できます。例えば、[Redis 固有のパラメータ](#) を使用してキャッシュエンジンを設定できます。
- 垂直方向と水平方向のスケール: 必要に応じてキャッシュノードのサイズを増減することで、クラスターを手動でスケールできます。新しいシャードを追加したり、シャードにレプリカを追加したりして、水平方向にスケールすることもできます。Auto-Scaling 機能を使用して、スケジュールに基づいてスケーリングを設定したり、キャッシュの CPU やメモリの使用状況などのメトリクスに基づいてスケーリングを設定することもできます。

次の図は、ElastiCache 自己設計クラスターの仕組みを示しています。



料金ディメンション

は 2 つのデプロイオプション ElastiCache でデプロイできます。ElastiCache Serverless をデプロイする場合、GB 時間単位で保存され、ElastiCache 処理ユニット (ECPU) で計算されたデータの使用料が発生します。Redis クラスター ElastiCache 用に独自の を設計することを選択する場合、キャッシュノードの使用料を 1 時間ごとに支払います。料金の詳細については、[こちら](#)を参照してください。

データストレージ

ElastiCache Serverless に保存されているデータの料金は、ギガバイト時間 (GB-hrs) 単位で請求されます。ElastiCache Serverless は、キャッシュに保存されているデータを継続的にモニタリングし、1 分間に複数回サンプリングし、1 時間あたりの平均を計算して、キャッシュのデータストレージの使用量を GB 時間単位で決定します。各 ElastiCache サーバーレスキャッシュは、最低 1 GB のデータが保存されるよう計測されます。

ElastiCache 処理ユニット (ECPUs)

アプリケーションが ElastiCache サーバーレスで実行する Redis リクエストに対して、vCPU 時間とデータ転送の両方を含むユニットである ElastiCache 処理ユニット (ECPUs) で料金を支払います。

- 単純な読み取りと書き込みには、転送されるデータの 1 キロバイト (KB) につき 1 ECPU が必要です。例えば、最大 1 KB のデータを転送する GET コマンドは 1 ECPU を消費します。3.2 KB のデータを転送する SET リクエストは 3.2 ECPU を消費します。
- 追加の vCPU 時間を必要とするコマンドは、それに比例して消費する ECPU が増えます。例えば、アプリケーションが Redis [HMGET コマンド](#) を使用し、単純な SET/GET コマンドの 3 倍の vCPU 時間を消費する場合、3 ECPU を消費することになります。
- vCPU 時間を多く消費し、より多くのデータを転送するコマンドは、2 つのディメンションのうち大きい方に基づいて消費する ECPU が決まります。例えば、アプリケーションが HMGET コマンドを使用して単純な SET/GET コマンドの 3 倍の vCPU 時間を消費し、3.2 KB のデータを転送する場合、3.2 ECPU を消費することになります。あるいは、2 KB のデータしか転送しない場合、3 ECPU を消費することになります。

ElastiCache Serverless は、ワークロードが消費する ElastiCache Processing Units (ECPUs) を理解するのに役立つという新しいメトリクスを発行します。

ノード時間

EC2 ノードファミリー、サイズ、ノード数、アベイラビリティーゾーン間の配置を選択することで、独自の Redis キャッシュクラスターを設計できます。クラスターを独自に設計する場合は、キャッシュノードごとに時間単位で料金を支払います。

ElastiCache Redis バックアップ用の

バックアップは Redis キャッシュ point-in-time のコピーです。ElastiCache を使用すると、いつでもデータのバックアップを作成したり、自動バックアップを設定したりできます。バックアップは、既存のキャッシュを復元するか、または新しいキャッシュをシードするのに使用できます。バックアップは、クラスターのすべてのデータといくつかのメタデータで構成されます。詳細については、「[スナップショットおよび復元](#)」を参照してください。

デプロイオプションの選択

Amazon ElastiCache には 2 つのデプロイオプションがあります。

- サーバーレスキャッシュ

• 独自設計型クラスター

両方でサポートされているコマンドのリストについては、「」を参照してください[サポートされている Redis コマンドと制限されている Redis コマンド](#)。

サーバーレスキャッシュ

Amazon ElastiCache Serverless はキャッシュの作成を簡素化し、お客様の最も要求の厳しいアプリケーションをサポートするために即座にスケールリングします。ElastiCache Serverless を使用すると、可用性が高くスケラブルなキャッシュを 1 分以内に作成できるため、キャッシュクラスターの容量をプロビジョニング、計画、管理する必要がありません。ElastiCache Serverless は 3 つの Availability Zones 間でデータを自動的に冗長的に保存し、99.99% の可用性サービスレベルアグリーメント (SLA) を提供します。バックアップは相互互換性があり、セルフデザインクラスターにエクスポートしたり、クラスターから復元したりできます。

独自設計型クラスター

ElastiCache for Redis クラスターをきめ細かく制御する必要がある場合は、を使用して独自の Redis クラスターを設計できます ElastiCache。ElastiCache では、クラスターのア AWS Availability Zones 全体でノードタイプ、ノード数、ノード配置を選択することで、ノードベースのクラスターを運用できます。ElastiCache はフルマネージドサービスであるため、クラスターのハードウェアプロビジョニング、モニタリング、ノード交換、およびソフトウェアパッチの管理に役立ちます。自己設計クラスターは、最大 99.99% の可用性 SLA を提供するように設計されています。バックアップは相互互換性があり、サービスレスキャッシュにエクスポートしたり、そこから復元したりできます。

デプロイオプションの選択

次の場合はサーバーレスキャッシュを選択してください。

- 新規または予測が難しいワークロードのキャッシュを作成している。
- アプリケーションのトラフィックが予測不可能です。
- キャッシュの使用を開始する最も簡単な方法が必要な場合。

次の場合は、独自の ElastiCache クラスターを設計することを選択します。

- すでに ElastiCache Serverless を実行しており、Redis を実行しているノードのタイプ、ノードの数、ノードの配置をより細かく制御したいと考えています。

- アプリケーショントラフィックは比較的予測可能であり、パフォーマンス、可用性、コストをきめ細かく制御する必要があります。
- キャパシティーの要件を予測してコストを管理できる場合。

サーバーレスキャッシュと自己設計クラスターの比較

機能	サーバーレスキャッシュ	独自設計型クラスター
キャッシュ設定	1分以内に名前だけを含むキャッシュを作成する	キャッシュクラスター設計をきめ細かく制御できます。ユーザーがノードタイプ、ノード数、ア AWS ベイラビリティゾーン間の配置を選択できる
Redis バージョン ElastiCache でサポートされる	ElastiCache for Redis バージョン 7.1 以降	ElastiCache for Redis バージョン 4.0 以降
クラスターモード	Redis は cluster mode enabledでのみ動作しません。Redis クライアントは、ElastiCache サーバーレスに接続cluster mode enabledするために をサポートする必要があります。	クラスターモードが有効または無効で動作するように設定できます。
スケーリング	容量管理なしで、垂直方向と水平方向の両方を自動的にスケーリングします。	スケーリングを制御すると同時に、現在の容量が需要に適切に対応していることを確認するためのモニタリングも必要です。 必要に応じてキャッシュノードのサイズを増減することで、垂直方向にスケーリングを選択できます。新しいシャードを追加するか、

機能	サーバーレスキャッシュ	独自設計型クラスター
		<p>シャードにレプリカを追加することで、水平方向にスケールすることもできます。</p> <p>自動スケーリング機能を使用すると、スケジュールに基づいてスケーリングを設定したり、キャッシュの CPU やメモリの使用状況などのメトリクスに基づいてスケーリングしたりすることもできます。</p>
クライアント接続	<p>クライアントは 1 つのエンドポイントに接続します。これにより、基盤となるキャッシュノードトポロジ（スケーリング、置換、アップグレード）をクライアントを切断せずに変更できます。</p>	<p>クライアントは個々のキャッシュノードに接続します。ノードが置き換えられると、クライアントはクラスタートポロジを再検出し、接続を再確立します。</p>
設定可能性	<p>きめ細かな設定は使用できません。お客様は、キャッシュにアクセスできるサブネット、自動バックアップがオンまたはオフになっているかどうか、キャッシュの最大使用制限など、基本的な設定を行うことができます。</p>	<p>自己設計クラスターは、きめ細かな設定オプションを提供します。お客様は、きめ細かな制御にパラメータグループを使用できます。ノードタイプ別のパラメータ値のテーブルについては、「Redis のノードタイプ固有のパラメータ」を参照してください。</p>

機能	サーバーレスキャッシュ	独自設計型クラスター
マルチ AZ	データは複数のアベイラビリティゾーンに非同期でレプリケートされるため、可用性が向上し、読み取りレイテンシーが向上します。	1つのアベイラビリティゾーンまたは複数のアベイラビリティゾーン (AZs)。マルチ AZ クラスターの場合、可用性を高め、読み取りレイテンシーを改善するために、データは複数のアベイラビリティゾーン間で非同期的にレプリケートされます。
保管中の暗号化	常に有効になっています。お客様は、 こちら で AWS マネージドキー またはカスタマーマネージドキーを使用できます AWS KMS。	保管時の暗号化を有効または無効にするオプション。有効にすると、 こちら で AWS マネージドキー またはカスタマーマネージドキーを使用できます AWS KMS。
転送時の暗号化 (TLS)	常に有効になっています。クライアントは TLS 接続をサポートしている必要があります。	有効または無効にするオプション。

機能	サーバーレスキャッシュ	独自設計型クラスター
バックアップ	<p>パフォーマンスに影響を与えずに、キャッシュの自動バックアップと手動バックアップをサポートします。</p> <p>バックアップには互換性があり、ElastiCache サーバーレスキャッシュまたは自己設計クラスターに復元できます。</p>	<p>自動バックアップと手動バックアップをサポートします。クラスターは、使用可能な予約メモリによっては、パフォーマンスに何らかの影響を与える可能性があります。詳細については、「予約メモリの管理」を参照してください。</p> <p>バックアップには互換性があり、ElastiCache サーバーレスキャッシュまたは自己設計クラスターに復元できます。</p>
モニタリング	<p>キャッシュヒット率、キャッシュミス率、データサイズ、消費ECPUsなどのキャッシュレベルのメトリクスをサポートします。</p> <p>ElastiCache Serverless は、キャッシュで重要なイベントが発生した EventBridge ときに、を使用してイベントを送信します。Amazon を使用して、ElastiCache イベントのモニタリング、取り込み、変換、および対応を選択できます EventBridge。詳細については、「サーバーレスキャッシュイベント」を参照してください。</p>	<p>ElastiCache 自己設計されたクラスターは、ホストレベルのメトリクスとキャッシュメトリクスの両方を含め、各ノードレベルでメトリクスを発行します。</p> <p>自己設計クラスターは、重要なイベントに対して SNS 通知を送信します。Redis のメトリクス を参照してください。</p>

機能	サーバーレスキャッシュ	独自設計型クラスター
可用性	99.99% 可用性 サービスレベルアグリーメント (SLA)	自己設計クラスターは、設定に応じて、最大 99.99% の可用性 サービスレベルアグリーメント (SLA) を実現できるように設計されています。
ソフトウェアのアップグレードとパッチ適用	アプリケーションに影響を与えることなく、キャッシュソフトウェアを最新のマイナーバージョンとパッチバージョンに自動的にアップグレードします。メジャーバージョンのアップグレードに関する通知が届きます。お客様は、必要に応じて最新のメジャーバージョンにアップグレードできます。	自己設計クラスターは、マイナーバージョンとパッチ適用バージョンのアップグレード、およびメジャーバージョンのアップグレードのために、お客様が有効にしたセルフサービスを提供します。マネージド更新は、お客様が定義したメンテナンスウィンドウ中に自動的に適用されます。お客様は、マイナーバージョンまたはパッチバージョンのアップグレードをオンデマンドで適用することもできます。
グローバルデータストア	サポートされていません	グローバルデータストアをサポート。単一リージョンの書き込みとマルチリージョンの読み取りによるクロスリージョンレプリケーションを可能にします。

機能	サーバーレスキャッシュ	独自設計型クラスター
データ階層化	サポートされていません	r6gd ファミリーのノードを使用して設計されたクラスターは、メモリとローカル SSD (ソリッドステートドライブ) ストレージの間でデータを階層化します。データ階層化は、データをメモリに保存することに加えて、各クラスターノードで低コストのソリッドステートドライブ (SSDs) を利用することで、Redis ワークロードにコストパフォーマンスの高いオプションを提供します。
料金モデル	P は ay-per-use、GB 時間で保存されたデータと ElastiCache 処理ユニット (ECPU) でのリクエストに基づいています。料金の詳細については、 こちら を参照してください。	キャッシュノードの使用状況 ay-per-hour に基づく P。料金の詳細については、 こちら を参照してください。

関連トピック:

- [独自の ElastiCache クラスターの設計と管理](#)

Amazon ElastiCache のリソース

以下のセクションを読んでから開始することをお勧めします。また、必要に応じて参照し直してください。

- [サービスのハイライトと価格設定] – 「[製品詳細ページ](#)」には、ElastiCache の全般的な製品概要、サービスのハイライト、料金表が掲載されています。

- [ElastiCache 動画] – 「[ElastiCache の動画](#)」セクションには Amazon ElastiCache を紹介する動画があります。この動画には、一般的ユースケースと、ElastiCache を使用してレイテンシーを減らし、アプリケーションのスループットを向上させる方法のデモがあります。
- [開始方法] – 「[Amazon ElastiCache for Redis を使い始める](#)」セクションには、キャッシュクラスタの作成に関する情報が記載されています。キャッシュクラスタへのアクセスを承認する方法、キャッシュノードに接続する方法、およびキャッシュクラスタを削除する方法も記載されています。
- [スケールに応じたパフォーマンス] – 「[Amazon ElastiCache を使用したスケールに応じたパフォーマンス](#)」ホワイトペーパーでは、アプリケーションがスケールに応じて適切に機能するためのキャッシュ戦略が紹介されています。

AWS Command Line Interface (AWS CLI) を使用する場合は、これらのドキュメントを使用すると、使用開始に役立ちます。

- [「AWS Command Line Interface ドキュメント」](#)

このセクションには、AWS CLI のダウンロード、システムでの AWS CLI の実行、AWS 認証情報の指定に関する情報が記載されています。

- [AWS CLI ElastiCache のドキュメント](#)

この別個のドキュメントは、構文と例などを含むすべての AWS CLI for ElastiCache コマンドについて説明します。

一般に使用されているさまざまなプログラミング言語を使用して、ElastiCache API を使用するアプリケーションプログラムを記述できます。次にいくつかのリソースを示します。

- [Amazon Web Services のツール](#)

Amazon Web Services には、ElastiCache をサポートする多数のソフトウェア開発キット (SDK) が用意されています。ElastiCache のコードは、Java、.NET、PHP、Ruby、および他の言語を使用できます。これらの SDK では、リクエストが ElastiCache の形式に設定されて、レスポンスが解析され、再試行ロジックとエラー処理が提供されるため、アプリケーション開発が大幅に簡略化されます。

- [ElastiCache API の使用](#)

AWS SDK を使用しない場合、クエリ API を使用して ElastiCache を直接操作することができます。リクエストを作成および認証してレスポンスを処理する際のトラブルシューティングのヒントと情報をこのセクションで確認できます。

- [Amazon ElastiCache API リファレンス](#)

この別個のドキュメントでは、構文と例などを含む、ElastiCache API オペレーションに関するすべてを説明します。

AWS リージョンとアベイラビリティゾーン

Amazon クラウドコンピューティングリソースは、世界各地 (例えば、北米、ヨーロッパ、アジア) の高可用性の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、AWS リージョンと呼ばれます。

各 AWS リージョンは、アベイラビリティゾーンまたは AZ と呼ばれる複数の区切られた場所で構成されています。各アベイラビリティゾーンは、他のアベイラビリティゾーンの障害から分離されるように設計されています。アベイラビリティゾーンは、同じ AWS リージョン内の他のアベイラビリティゾーンに低価格かつ低レイテンシーのネットワーク接続を提供します。個別のアベイラビリティゾーンでインスタンスを起動することにより、1つの場所で発生した障害からアプリケーションを保護できます。詳細については、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

オプションで、マルチ AZ 配置と呼ばれる複数のアベイラビリティゾーンのクラスターを作成できます。このオプションを選択すると、Amazon によってセカンダリスタンバイノードインスタンスが別のアベイラビリティゾーンで自動的にプロビジョンされて管理されます。プライマリインスタンスは、非同期でアベイラビリティゾーン間でセカンダリインスタンスにレプリケートされます。これは、データの冗長性およびフェイルオーバーサポートを提供し、I/O フリーズを排除して、システムのバックアップの際のレイテンシーのスパイクを最小限に抑えるのに役立ちます。詳細については、「[マルチ AZ を用いた ElastiCache for Redis でのダウンタイムの最小化](#)」を参照してください。

ElastiCache の一般的なユースケースおよび ElastiCache がどのように役立つか

最新のニュース、トップ 10 のリーダーボード、製品カタログ、またはイベントのチケットを販売できます。スピードの勝負です。ウェブサイトやビジネスの成功は、コンテンツを配信するスピードに大きく左右されます。

New York Times の「[For Impatient Web Users, an Eye Blink Is Just Too Long to Wait](#)」によると、ユーザーは競合サイト間で 250 ミリ秒 (1/4 秒) の違いを認識します。ユーザーは、遅いサイトより速度の速いサイトのほうを選びます。Amazon が行ったテスト「[How Webpage Load Time Is Related to Visitor Loss](#)」では、ロード時間が 100 ミリ秒 (1/10 秒) 長くなるごとに、売上げが 1 パーセント減少するとの結果が出ています。

ある人がデータを必要とする場合、そのデータをキャッシュしておくことで、より速く配信できます。ウェブページであろうとビジネスの意思決定にかかわるレポートであろうと、これは事実です。ビジネス上、できる限り短いレイテンシーでウェブページを配信するためにウェブページをキャッシュしないでいられることがあるのでしょうか。

最も頻繁にリクエストされる項目をキャッシュするべきであることは、考えるまでもなく明白でしょう。しかし、リクエスト頻度の低い項目をキャッシュしない方がいいのでしょうか。最も最適化されたデータベースクエリまたはリモート API コールを使用しても、インメモリキャッシュからの取得に比べれば、著しく時間がかかります。著しく時間がかかると、顧客を他社に取られてしまいます。

次の例で、ElastiCache を使用してアプリケーションの全体的なパフォーマンスを向上させるいくつかの方法を説明します。

トピック

- [インメモリデータストア](#)
- [ゲームのリーダーボード \(Redis ソートセット\)](#)
- [メッセージング \(Redis パブリッシュ/サブスクライブ\)](#)
- [推奨データ \(Redis ハッシュ\)](#)
- [その他の Redis の用途](#)
- [ElastiCache のお客様の声](#)

インメモリデータストア

インメモリキー値ストアの主な目的は、データのコピーに超高速 (ミリ秒以下のレイテンシー) で低コストなアクセスを提供することです。ほとんどのデータストアには、頻繁にアクセスされてもほとんど更新されることのないデータ領域があります。さらにデータベースのクエリは、キーと値のペアのキャッシュを検索するよりも常に時間がかかり、キーの検索にコストがかかります。一部のデータベースクエリは、実行に特にコストがかかります。例として、複数のテーブルにまたがるクエリや、集中的な計算が必要なクエリが挙げられます。このようなクエリの結果をキャッシュすることで、クエリの価格が一度だけ支払われることになります。その後、クエリを再実行することなく、データを何回でもすぐに取得できます。

キャッシュの方法。

キャッシュするデータを決める場合は、以下の点を考慮してください。

[速度とコスト] – データベースからデータを取得するには、キャッシュから取得するより常に時間とコストがかかります。データベースのクエリによっては、本質的により低速で高コストのものもあります。たとえば、複数のテーブルにわたって実行されるクエリは、単純な単一テーブルに対するクエリよりもコストが高くなります。興味深いデータを取得するのに時間のかかるコストの高いクエリが必要となるのであれば、キャッシュを検討する価値があります。データを比較的単純なクエリで迅速に取得できる場合であっても、その他の要因によってはキャッシュを検討する価値があります。

[データとアクセスパターン] – キャッシュするデータを決定するには、データ自体とアクセスパターンを理解することも必要です。たとえば、すぐに変化するデータやほとんどアクセスされることのないデータをキャッシュすることには意味がありません。キャッシュに有意な利点を持たせるには、データは比較的静的で頻繁にアクセスされる必要があります。例としては、ソーシャルメディアサイト上の個人プロフィールがあります。一方、キャッシュによる速度またはコストのメリットがない場合は、データをキャッシュする意味はありません。たとえば、検索結果を返すウェブページをキャッシュすることは、クエリと結果は通常固有のものであるため、意味がありません。

[古さ] – 定義上、キャッシュされたデータは古いデータです。たとえ特定の状況でそれが古くなっていなくても、それは常に古くなっているとみなされ、そのように扱われるべきです。データがキャッシュの候補となりうるかどうかを確認する際に、古いデータに対するアプリケーションの耐障害性を判断します。

アプリケーションでは、あるコンテキストで古いデータを許容できても、別のコンテキストでは許容できない場合があります。たとえば、サイトが公開されている株価を提供しているとします。あなたの顧客は、価格が n 分遅れているかもしれないという免責条項で何らかの古さを受け入れる場合が

あります。しかし、売買を行うブローカーにその株価を提供する場合は、リアルタイムのデータが必要になります。

次のことが成り立つ場合、データをキャッシュすることを検討します。

- また、キャッシュの取得に比べて、データは遅く、コストがかかります。
- ユーザーは頻繁にデータにアクセスします。
- データは比較的同じままであるか、データが急速に変化しても、古さは大きな問題ではありません。

詳細については、次を参照してください:

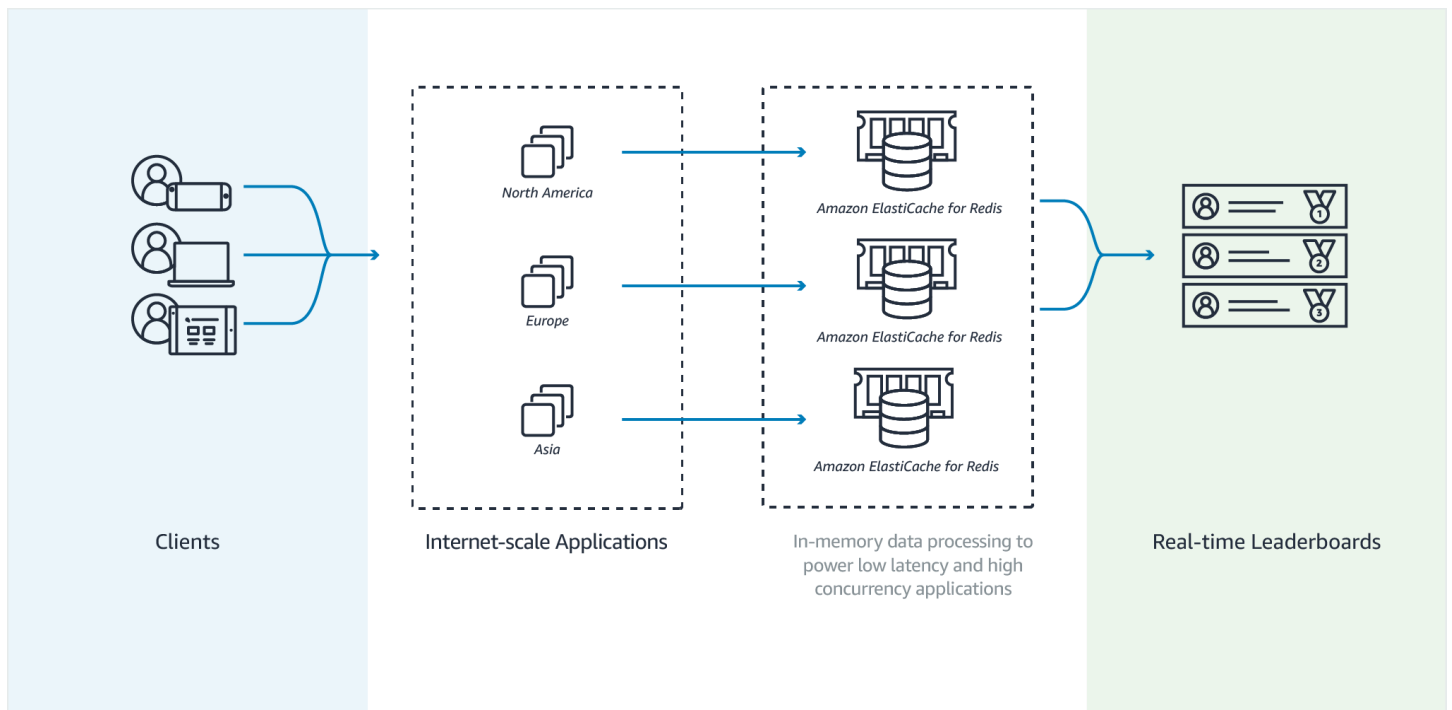
- ElastiCache for Redis ユーザーガイドの「[キャッシュ戦略](#)」

ゲームのリーダーボード (Redis ソートセット)

Redis ソートセットは、リーダーボードの計算の複雑性を、アプリケーションから Redis クラスターへ移動します。

ゲームのハイスコアのトップ 10 などのリーダーボードは計算が複雑です。これは、大勢のプレイヤーが同時にプレーしており、スコアが継続的に変化する場合は、特に当てはまります。Redis ソートセットは、一意性と要素の順番を保証します。Redis ソートセットを使用すると、ソートセットに新しい要素が追加されるたびに、リアルタイムで再ランキングが行われます。次にそれは、正しい番号順でソートセットに追加されます。

次の図を見ると、ElastiCache for Redis のゲームのリーダーボードがどのように動作するかがわかります。



Example - Redis リーダーボード

この例では、ZADD を使用して 4 人のゲーマーとそのスコアがソートされたリストに入力されています。ZREVRANGEBYSCORE コマンドは、プレイヤーをスコアの高い者から順に一覧します。次に、ZADD を使用して既存のエントリを上書きして、June のスコアを更新します。最後に、ZREVRANGEBYSCORE コマンドは、プレイヤーをスコアの高い者から順に一覧します。リストは、June のランキングが上昇したことを示しています。

```
ZADD leaderboard 132 Robert
ZADD leaderboard 231 Sandra
ZADD leaderboard 32 June
ZADD leaderboard 381 Adam
```

```
ZREVRANGEBYSCORE leaderboard +inf -inf
```

- 1) Adam
- 2) Sandra
- 3) Robert
- 4) June

```
ZADD leaderboard 232 June
```

```
ZREVRANGEBYSCORE leaderboard +inf -inf
```

- 1) Adam
- 2) June

- 3) Sandra
- 4) Robert

次のコマンドは、June にすべてのプレイヤー間での自分のランクを知らせます。ランク付けがゼロベースであるため、ZREVRANK は 2 番目の位置にいる June に対して 1 を返します。

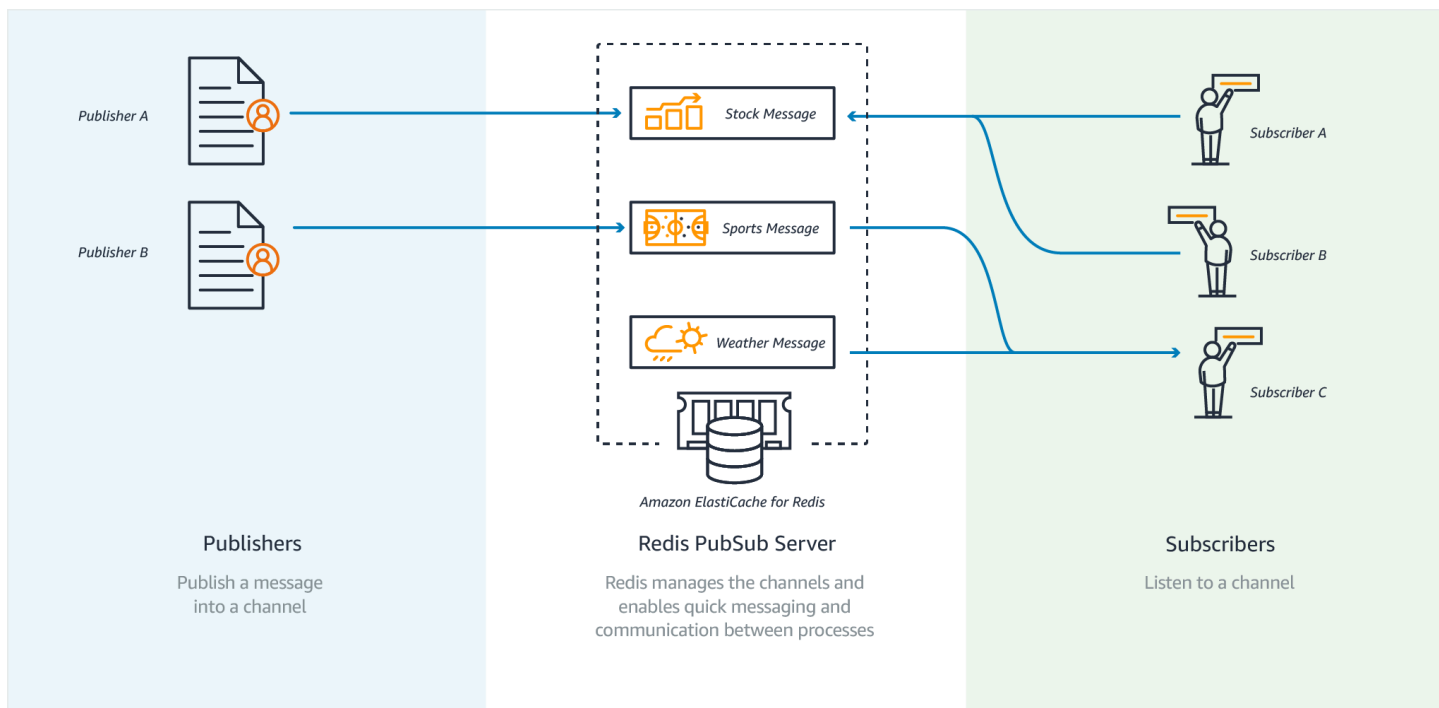
```
ZREVRANK leaderboard June
1
```

詳細については、ソートセットについての「[Redis のドキュメント](#)」を参照してください。

メッセージング (Redis パブリッシュ/サブスクライブ)

E メールメッセージを送信すると、1 人以上の指定された受信者にメッセージが送信されます。publish/subscribe のパラダイムでは、メッセージをその受信者にではなく、受信者を特定しないまま特定のチャンネルに送信します。メッセージを受け取る人は、そのチャンネルに登録している人です。たとえば、お客様が news.sports.golf チャンネルに登録しているとします。news.sports.golf へのすべての登録者は、news.sports.golf チャンネルに発行されるメッセージをすべて受け取ります。

Redis の publish/subscribe 機能は、キー空間とは無関係です。したがって、あらゆるレベルで干渉されることはありません。次の図は、ElastiCache for Redis のメッセージングを示したものです。



登録中

チャンネルに発行されるメッセージを受け取るには、チャンネルに登録してください。1つのチャンネル、複数の指定されたチャンネル、またはパターンに一致するすべてのチャンネルに登録できます。登録を取り消すには、登録時に指定したチャンネルから登録解除します。または、パターンマッチングを使用して登録した場合は、以前に使用したのと同じパターンを使用して登録解除します。

Example - 1つのチャンネルへの登録

1つのチャンネルに登録するには、登録するチャンネルを指定して SUBSCRIBE コマンドを使用します。以下の例では、クライアントは news.sports.golf チャンネルに登録します。

```
SUBSCRIBE news.sports.golf
```

しばらくすると、クライアントは、登録解除するチャンネルを指定した UNSUBSCRIBE コマンドを使用して、チャンネルへの登録をキャンセルします。

```
UNSUBSCRIBE news.sports.golf
```

Example - 複数の指定されたチャンネルへの登録

複数の指定されたチャンネルに登録するには、SUBSCRIBE コマンドを使用してチャンネルに登録します。以下の例では、クライアントは news.sports.golf、news.sports.soccer、news.sports.skiing のすべてのチャンネルに登録します。

```
SUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

特定のチャンネルへの登録をキャンセルするには、UNSUBSCRIBE コマンドを使用して、登録解除するチャンネルを指定します。

```
UNSUBSCRIBE news.sports.golf
```

複数のチャンネルへの登録をキャンセルするには、UNSUBSCRIBE コマンドを使用して、登録解除するチャンネルを指定します。

```
UNSUBSCRIBE news.sports.golf news.sports.soccer
```

すべての登録をキャンセルするには、UNSUBSCRIBE を使用して、各チャンネルを指定します。または、UNSUBSCRIBE を使用して、チャンネルを指定しないでください。

```
UNSUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

または

```
UNSUBSCRIBE
```

Example - パターンマッチングを使用した登録

クライアントは PSUBSCRIBE コマンドを使用して、パターンに一致するすべてのチャンネルに登録できます。

以下の例では、クライアントはすべてのチャンネルに登録します。SUBSCRIBE を使用して行うように、すべてのスポーツチャンネルを個別にリストしないでください。代わりに、PSUBSCRIBE コマンドでは、パターンマッチングを使用します。

```
PSUBSCRIBE news.sports.*
```

Example 登録のキャンセル

これらのチャンネルへの登録をキャンセルするには、PUNSUBSCRIBE コマンドを使用します。

```
PUNSUBSCRIBE news.sports.*
```

Important

[P]SUBSCRIBE コマンドに送られるチャンネルの文字列と、[P]UNSUBSCRIBE コマンドに送られるチャンネルの文字列は一致している必要があります。PSUBSCRIBE を news.* に、また PUNSUBSCRIBE を news.sports.* から、または UNSUBSCRIBE を news.sports.golf から行うことはできません。

公開

チャンネルへのすべての登録者にメッセージを送信するには、PUBLISH コマンドを使用してチャンネルとメッセージを指定します。以下の例では、"It's Saturday and sunny. I'm headed to the links." というメッセージを news.sports.golf チャンネルに発行しています。

```
PUBLISH news.sports.golf "It's Saturday and sunny. I'm headed to the links."
```

クライアントは、登録しているチャンネルに発行することはできません。

詳細については、『Redis ドキュメント』の「[Pub/Sub](#)」を参照してください。

推奨データ (Redis ハッシュ)

Redis で INCR または DECR を使用すると、コンパイルの推奨事項が簡単になります。ユーザーが製品を「好き」になるたびに、項目: productID: 好みに 1 を加えます。ユーザーが製品を「嫌い」になるたびに、項目: productID: 嫌いに 1 を加えます。Redis ハッシュを使用して、その製品を好きなまたは嫌いなユーザー全員のリストを保持できます。

Example - 好き/嫌い

```
INCR item:38923:likes
HSET item:38923:ratings Susan 1
INCR item:38923:dislikes
HSET item:38923:ratings Tommy -1
```

その他の Redis の用途

Salvatore Sanfilippo によるブログ投稿 ([How to take advantage of Redis just adding it to your stack](#)) では、数多くのよく知られたデータベースの懸念が、Redis を使用して問題を簡単に解決している方法が紹介されています。この方法では、データベースの負荷がなくなり、パフォーマンスが向上します。

ElastiCache のお客様の声

Airbnb、PBS、Esri、その他の企業が、Amazon ElastiCache を使用して自社の顧客体験を向上させている方法については、「[他のお客様が Amazon ElastiCache をどのように使用しているか](#)」を参照してください。

[チュートリアルビデオ](#)で ElastiCache のお客様の他のユースケースを見ることもできます。

Amazon ElastiCache for Redis を使い始める

このセクションのハンズオンチュートリアルを参考にして、ElastiCache for Redis の使用を開始し、詳細を学んでください。

トピック

- [セットアップ](#)
- [ステップ 1: キャッシュを作成する](#)
- [ステップ 2: キャッシュへのデータの読み取りと書き込みを行う](#)
- [ステップ 3: \(オプション\) クリーンアップする](#)
- [次のステップ](#)
- [ElastiCache および AWS SDK の使用開始](#)
- [チュートリアル: Amazon VPC ElastiCache 内のアマゾンにアクセスするための Lambda 関数の設定](#)

セットアップ

を設定するには ElastiCache :

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [プログラマ的なアクセス権を付与する](#)
- [アクセス許可を設定する \(新規 ElastiCache ユーザーのみ\)](#)
- [EC2 をセットアップする](#)
- [Amazon VPC セキュリティグループからキャッシュへのネットワークネットワークアクセスを許可する](#)
- [redis-cli をダウンロードしてセットアップする](#)

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

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

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

管理アクセスを持つユーザーを作成する

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

のセキュリティ保護 AWS アカウントのルートユーザー

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

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとしてサインインする](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

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

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

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

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

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

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

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

プログラムのなアクセス権を付与する

ユーザーが AWS 外部で操作する場合は、プログラムによるアクセスが必要です AWS Management Console。プログラムによるアクセスを許可する方法は、にアクセスするユーザーのタイプによって異なります AWS。

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

プログラマチックアクセス権を必要とするユーザー	目的	方法
<p>ワークフォースアイデンティティ</p> <p>(IAM Identity Center で管理されているユーザー)</p>	<p>一時的な認証情報を使用して、AWS SDKs、AWS CLI、または AWS APIs。</p>	<p>使用するインターフェイス用の手引きに従ってください。</p> <ul style="list-style-type: none"> • については AWS CLI、「ユーザーガイド」の AWS CLI「を使用するための設定 AWS IAM Identity Center AWS Command Line Interface」を参照してください。 • AWS SDKs、ツール、AWS APIs「SDK とツールのリファレンスガイド」の 「IAM Identity Center 認証」を参照してください。 AWS SDKs
IAM	<p>一時的な認証情報を使用して、AWS SDKs、AWS CLI、または AWS APIs。</p>	<p>「IAM ユーザーガイド」の 「AWS リソースでの一時的な認証情報の使用」の手順に従います。</p>
IAM	<p>(非推奨)</p> <p>長期認証情報を使用して、AWS SDKs、AWS CLI、または AWS APIs。</p>	<p>使用するインターフェイス用の手引きに従ってください。</p> <ul style="list-style-type: none"> • については AWS CLI、「AWS Command Line Interface ユーザーガイド」の 「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDKs「SDK とツールのリファレンスガイド」の 「長期的な認証情報を使

プログラマチックアクセス権を必要とするユーザー	目的	方法
		<p>用した認証」を参照してください。AWS SDKs</p> <ul style="list-style-type: none"> • AWS APIs ユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

関連トピック:

- IAM ユーザーガイドの [IAM とは](#)
- AWS 全般のリファレンス [AWS の「セキュリティ認証情報](#)」。

アクセス許可を設定する (新規 ElastiCache ユーザーのみ)

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

- のユーザーとグループ AWS IAM Identity Center :

アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「[権限設定を作成する](#)」の手順に従ってください。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

- IAM ユーザー:

- ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「[IAM ユーザー用ロールの作成](#)」を参照してください。
- (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「[ユーザー \(コンソール\) へのアクセス権限の追加](#)」を参照してください。

Amazon は、ユーザーに代わってリソースをプロビジョニングし、他の AWS リソースやサービスにアクセスするために、サービスにリンクされたロール ElastiCache を作成および使用します。

ElastiCache でサービスにリンクされたロールを作成するには、という名前の AWS マネージドポリシーを使用します `AmazonElastiCacheFullAccess`。このロールには、サービスにリンクされたロールをサービスがユーザーに代わって作成するために必要なアクセス許可が事前に設定されています。

デフォルトのポリシーを使用せず、代わりにカスタム管理ポリシーを使用することもできます。この場合、 を呼び出すアクセス許可を持っているか、サービスにリンクされたロールを作成 `iam:createServiceLinkedRole` ElastiCache 済みであることを確認してください。

詳細については、次を参照してください。

- [新しいポリシーの作成 \(IAM\)](#)
- [Amazon ElastiCache の AWS マネージドポリシー](#)
- [Amazon ElastiCache でのサービスにリンクされたロールの使用](#)

EC2 をセットアップする

キャッシュに接続する EC2 インスタンスを設定する必要があります。

- EC2 インスタンスがまだない場合は、「[Amazon EC2 Linux インスタンスの開始方法](#)」で EC2 インスタンスのセットアップ方法を参照してください。
- EC2 インスタンスはキャッシュと同じ VPC にあり、同じセキュリティグループ設定を持っている必要があります。デフォルトでは、Amazon ElastiCache はデフォルトの VPC にキャッシュを作成し、デフォルトのセキュリティグループを使用します。このチュートリアルを進めるには、EC2 インスタンスがデフォルト VPC にあり、デフォルトのセキュリティグループが設定されていることを確認してください。

Amazon VPC セキュリティグループからキャッシュへのネットワークネットワークアクセスを許可する

ElastiCache 独自設計型クラスターは Redis コマンドにポート 6379 を使用し、ElastiCache サーバーレスはポート 6379 とポート 6380 の両方を使用します。EC2 インスタンスから Redis コマンドを正常に接続して実行するには、セキュリティグループが必要に応じてこれらのポートへのアクセスを許可する必要があります。

1. にサインイン AWS Command Line Interface し、[Amazon EC2 コンソール](#) を開きます。

2. ナビゲーションペインで、[ネットワーク & セキュリティ] の下にある [セキュリティグループ] を選択します。
3. セキュリティグループのリストから、Amazon VPC のセキュリティグループを選択します。ElastiCache 使用するセキュリティグループを作成しない限り、このセキュリティグループにはデフォルトの という名前が付けられます。
4. [インバウンド] タブを開き、[編集] をクリックします。
 - a. Edit (編集) を選択します。
 - b. ルールの追加 を選択します。
 - c. [タイプ] 列で [カスタム TCP ルール] を選択します。
 - d. [ポート範囲] ボックスに、6379 と入力します。
 - e. [送信元] ボックスで [任意の場所] を選択します。ポート範囲が 0.0.0.0/0 になるため、Amazon VPC 内で起動したすべての Amazon EC2 インスタンスをキャッシュに接続できます。
 - f. ElastiCache サーバーレスを使用している場合は、ルールの追加 を選択して別のルールを追加します。
 - g. Type 列で Custom TCP rule を選択します。
 - h. [ポート範囲] ボックスに、6380 と入力します。
 - i. [送信元] ボックスで [任意の場所] を選択します。ポート範囲が 0.0.0.0/0 になるため、Amazon VPC 内で起動したすべての Amazon EC2 インスタンスをキャッシュに接続できます。
 - j. [保存] を選択します。

redis-cli をダウンロードしてセットアップする

1. 選択した接続ユーティリティを使用して、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方法については、「[Amazon EC2 入門ガイド](#)」を参照してください。
2. セットアップに適したコマンドを実行し、redis-cli ユーティリティをダウンロードしてインストールします。

Amazon Linux 2023

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

Amazon Linux 2

```
sudo amazon-linux-extras install epel -y
sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y
sudo wget http://download.redis.io/redis-stable.tar.gz
sudo tar xvzf redis-stable.tar.gz
cd redis-stable
sudo make BUILD_TLS=yes
```

Note

- redis6 パッケージをインストールすると、デフォルトの暗号化サポートで redis6-cli がインストールされます。
- redis-cli をインストールするときは、TLS のビルドサポートを受けることが重要です。ElastiCache Serverless は、TLS が有効になっている場合にのみアクセスできます。
- 接続先のクラスターが暗号化されていない場合、Build_TLS=yes オプションは必要ありません。

ステップ 1: キャッシュを作成する

このステップでは、Amazon ElastiCache に新しいキャッシュを作成します。

AWS Management Console

ElastiCache コンソールを使用して新しいキャッシュを作成するには:

1. AWS Management Consoleにサインインし、<https://console.aws.amazon.com/connect/> 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、[Redis キャッシュ] を選択します。
3. コンソールの右側で、[Redis キャッシュを作成] を選択します。
4. [キャッシュ設定] に [名前] を入力します。オプションで、キャッシュの [説明] を入力できます。
5. デフォルトの設定を選択したままにしておきます。
6. [作成] をクリックして、キャッシュを作成します。

7. キャッシュが「アクティブ」状態になったら、キャッシュへのデータの書き込みと読み取りを開始できます。

AWS CLI

以下の AWS CLI の例では、`create-serverless-cache` を使用して新しいキャッシュを作成します。

Linux

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name CacheName \  
  --engine redis
```

Windows

```
aws elasticache create-serverless-cache ^  
  --serverless-cache-name CacheName ^  
  --engine redis
```

ステータスフィールドの値が `CREATING` に設定されていることに注意してください。

ElastiCache がキャッシュの作成を終了したことを確認するには、`describe-serverless-caches` コマンドを使用します。

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

新しいキャッシュを作成したら、「[ステップ 2: キャッシュへのデータの読み取りと書き込みを行う](#)」に進みます。

ステップ 2: キャッシュへのデータの読み取りと書き込みを行う

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

また、このセクションでは、キャッシュに接続する EC2 インスタンスの VPC アクセスとセキュリティグループの設定が完了し、EC2 インスタンスに redis-cli が設定されていることを前提としています。このステップの詳細については、「[セットアップ](#)」を参照してください。

キャッシュエンドポイントを見つける

AWS Management Console

ElastiCache コンソールを使用してキャッシュのエンドポイントを検索するには：

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で Amazon ElastiCache 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、[Redis キャッシュ] を選択します。
3. コンソールの右側で、作成したキャッシュの名前をクリックします。
4. [キャッシュ詳細] で、キャッシュエンドポイントを見つけてコピーします。

AWS CLI

次の AWS CLI 例は、describe-serverless-caches コマンドを使用して新しいキャッシュのエンドポイントを検索する方法を示しています。コマンドを実行したら、「Endpoint」フィールドを探します。

Linux

```
aws elasticache describe-serverless-caches \  
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^  
  --serverless-cache-name CacheName
```

Redis キャッシュに接続する (Linux)

これで、必要なエンドポイントが手に入ったので、EC2 インスタンスにログインし、キャッシュに接続できます。次の例では、redis-cli ユーティリティを使用して、クラスターに接続します。次のコマンドでキャッシュに接続します (注: cache-endpoint は前のステップで取得したエンドポイントに置き換えてください)。


```
src/redis-cli -h cache-endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

Redis キャッシュに接続する (Windows)

これで、必要なエンドポイントが手に入ったので、EC2 インスタンスにログインし、キャッシュに接続できます。次の例では、redis-cli ユーティリティを使用して、クラスターに接続します。以下のコマンドでキャッシュに接続します。コマンドプロンプトを開いて Redis ディレクトリに移動し、コマンドを実行します (注: Cache_Endpoint は前のステップで取得したエンドポイントに置き換えてください)。

```
c:\Redis>redis-cli -h Redis_Cluster_Endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

これで、「[ステップ 3: \(オプション\) クリーンアップする](#)」に進むことができます。

ステップ 3: (オプション) クリーンアップする

作成した Amazon ElastiCache のキャッシュが不要になった場合は、削除できます。このステップにより、使用していないリソースに対して課金されることがなくなります。ElastiCache コンソール、AWS CLI、または ElastiCache API を使用してキャッシュを削除できます。

AWS Management Console

コンソールを使用してキャッシュを削除するには:

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) 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、[Redis キャッシュ] を選択します。
3. 削除するキャッシュの横にあるボタンを選択します。
4. 右上の [アクション] を選択し、[削除] を選択します。
5. オプションで、キャッシュを削除する前に最終スナップショットを取ることもできます。

6. [削除] 確認画面で、[削除] を選択してクラスターを削除するか、[キャンセル] を選択してクラスターを保持します。

キャッシュのステータスが [Deleting] になると、キャッシュの課金も停止されます。

AWS CLI

次の AWS CLI の例では、delete-serverless-cache コマンドを使用してキャッシュを削除します。

Linux

```
aws elasticache delete-serverless-cache \  
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache delete-serverless-cache ^  
  --serverless-cache-name CacheName
```

[ステータス] フィールドの値が [Deleting] になっていることに注意してください。

これで、「[次のステップ](#)」に進むことができます。

次のステップ

ElastiCache の詳細については、以下のウェブページを参照してください。

- [の使用 ElastiCache](#)
- [Redis ElastiCache のスケーリング](#)
- [Amazon ElastiCache でのログ記録とモニタリング](#)
- [ElastiCache ベストプラクティスとキャッシュ戦略](#)
- [スナップショットおよび復元](#)
- [Amazon SNS による ElastiCache イベントのモニタリング](#)

ElastiCache および AWS SDK の使用開始

このセクションには、Amazon ElastiCache の学習に役立つハンズオンチュートリアルが含まれています。言語固有のチュートリアルの 1 つを実行することをお勧めします。

Note

AWS SDK は、さまざまな言語で利用できます。詳細なリストについては、「[アマゾン ウェブ サービスのツール](#)」を参照してください。

Python と ElastiCache

このチュートリアルでは、AWS SDK for Python (Boto3) を使用して次の ElastiCache オペレーションを実行するシンプルなプログラムを記述します。

- ElastiCache クラスターを作成する (クラスターモードが有効およびクラスターモードが無効)
- ユーザーまたはユーザーグループが存在するかどうかを確認し、存在しない場合は作成する (Redis 6.0 以降のみ)
- ElastiCache に接続する
- 文字列の設定と取得、ストリームからの読み取りと書き込み、Pub/Sub チャンネルからの公開と登録などのオペレーションを実行します。

このチュートリアルは、AWS SDK for Python (Boto) ドキュメントを参照しながら実行できます。以下のセクションは [ElastiCache の低レベルのクライアント](#) に固有のものです。

チュートリアルの前提条件

- AWS SDK を使用するために必要な AWS アクセスキーをセットアップします。詳細については、「[セットアップ](#)」を参照してください。
- Python 3.0 以降をインストールします。詳細については、<https://www.python.org/downloads> を参照してください。手順については、Boto 3 ドキュメントの「[クイックスタート](#)」を参照してください。

ElastiCache クラスターとユーザーの作成

以下の例では、ElastiCache 管理操作 (クラスターまたはユーザー作成) には boto3 SDK を使用し、データ処理には redis-py-cluster redis-py/ を使用しています。

トピック

- [クラスターモードが無効のクラスターを作成する](#)

- [TLS と RBAC を用いてクラスターモードが無効のクラスターを作成する](#)
- [クラスターモードが有効のクラスターの作成](#)
- [TLS および RBAC を用いたクラスターモードが有効のクラスターの作成](#)
- [ユーザー/ユーザーグループが存在するかどうかを確認し、そうでない場合は作成する](#)

クラスターモードが無効のクラスターを作成する

次のプログラムをコピーして、.py という名前のファイルに貼り付けます。CreateClusterModeDisabledCluster

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
create_cluster_mode_disabled(CacheNodeType='cache.t3.small', EngineVersion='6.0', NumCacheClusters=1,
cache_cluster', ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode disabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
node) and maximum 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
```

```
        AutomaticFailoverEnabled=True,  
        CacheNodeType=CacheNodeType,  
        Engine='redis',  
        EngineVersion=EngineVersion,  
        NumCacheClusters=NumCacheClusters,  
        ReplicationGroupDescription=ReplicationGroupDescription,  
        ReplicationGroupId=ReplicationGroupId,  
        SnapshotRetentionLimit=30,  
    )  
    return response  
  
if __name__ == '__main__':  
  
    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large  
    nodes, Redis 6, one primary and two replicas  
    elasticacheResponse = create_cluster_mode_disabled(  
        #CacheNodeType='cache.m6g.large',  
        EngineVersion='6.0',  
        NumCacheClusters=3,  
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',  
        ReplicationGroupId='redis202104053'  
    )  
  
    logging.info(elasticacheResponse)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python CreateClusterModeDisabledCluster.py
```

詳細については、「[クラスターの管理](#)」を参照してください。

TLS と RBAC を用いてクラスターモードが無効のクラスターを作成する

セキュリティを確保するために、クラスターモードが無効のクラスターを作成するときに、Transport Layer Security (TLS) とロールベースのアクセスコントロール (RBAC) を使用できます。トークンが認証された場合に認証されたすべてのクライアントが完全なレプリケーショングループアクセスを持つ Redis AUTH とは異なり、RBAC ではユーザーグループを介してクラスターアクセスを制御できます。これらのユーザーグループは、レプリケーショングループへのアクセスを分類する方法として設計されています。詳細については、「[ロールベースのアクセスコントロール \(RBAC\)](#)」を参照してください。

次のプログラムをコピーして ClusterModeDisabledWithRBAC.py という名前のファイルに貼り付けます。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
create_cluster_mode_disabled_rbac(CacheNodeType='cache.t3.small', EngineVersion='6.0', NumCacheC
cache cluster', ReplicationGroupId=None, UserGroupIds=None,
SecurityGroupIds=None, CacheSubnetGroupName=None):
    """Creates an ElastiCache Cluster with cluster mode disabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
node) and maximum 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Mandatory name for the cluster.
    :param UserGroupIds: The ID of the user group to be assigned to the cluster.
    :param SecurityGroupIds: List of security groups to be assigned. If not defined,
default will be used
    :param CacheSubnetGroupName: subnet group where the cluster will be placed. If not
defined, default will be used.
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return {'Error': 'ReplicationGroupId parameter is required'}
    elif not isinstance(UserGroupIds, (list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
```

```
'Engine': 'redis',
'EngineVersion': EngineVersion,
'NumCacheClusters': NumCacheClusters,
'ReplicationGroupDescription': ReplicationGroupDescription,
'ReplicationGroupId': ReplicationGroupId,
'SnapshotRetentionLimit': 30,
'TransitEncryptionEnabled': True,
'UserGroupIds':UserGroupIds
}

# defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
if isinstance(SecurityGroupIds,(list)):
    params.update({'SecurityGroupIds':SecurityGroupIds})
if CacheSubnetGroupName:
    params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

response = client.create_replication_group(**params)
return response

if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
    nodes, Redis 6, one primary and two replicas.
    # Assigns the existent user group "mygroup" for RBAC authentication

    response=create_cluster_mode_disabled_rbac(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        NumCacheClusters=3,
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',
        ReplicationGroupId='redis202104',
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
        CacheSubnetGroupName='default'
    )

    logging.info(response)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ClusterModeDisabledWithRBAC.py
```

詳細については、「[クラスターの管理](#)」を参照してください。

クラスタモードが有効のクラスターの作成

次のプログラムをコピーして、ClusterModeEnabled.py という名前のファイルに貼り付けます。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
    create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
    ReplicationGroupDescription='Sample cache with cluster mode
    enabled',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode enabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
    cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
    CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
    used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximum 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
    replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='redis',
```



```
        EngineVersion=EngineVersion,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
        # Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
node (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=NumNodeGroups,
        ReplicasPerNodeGroup=ReplicasPerNodeGroup,
        CacheParameterGroupName='default.redis6.0.cluster.on'
    )

    return response

# Creates a cluster mode enabled
response = create_cluster_mode_enabled(
    CacheNodeType='cache.m6g.large',
    EngineVersion='6.0',
    ReplicationGroupDescription='Redis cluster mode enabled with replicas',
    ReplicationGroupId='redis20210',
    # Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
(implicit) and 2 replicas (replicasPerNodeGroup)
    NumNodeGroups=2,
    ReplicasPerNodeGroup=1,
)

logging.info(response)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ClusterModeEnabled.py
```

詳細については、「[クラスターの管理](#)」を参照してください。

TLS および RBAC を用いたクラスターモードが有効のクラスターの作成

セキュリティを確保するために、クラスターモードが有効のクラスターを作成するときに、Transport Layer Security (TLS) とロールベースのアクセスコントロール (RBAC) を使用できます。トークンが認証された場合に認証されたすべてのクライアントが完全なレプリケーショングループアクセスを持つ Redis AUTH とは異なり、RBAC ではユーザーグループを介してクラスターアクセスを制御できます。これらのユーザーグループは、レプリケーショングループへのアクセスを分類する方法として設計されています。詳細については、「[ロールベースのアクセスコントロール \(RBAC\)](#)」を参照してください。

次のプログラムをコピーして ClusterModeEnabledWithRBAC.py という名前のファイルに貼り付けます。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
    create_cluster_mode_enabled(CacheNodeType='cache.t3.small', EngineVersion='6.0', NumNodeGroups=1,
    ReplicationGroupDescription='Sample cache with cluster
    mode enabled', ReplicationGroupId=None, UserGroupIds=None,
    SecurityGroupIds=None, CacheSubnetGroupName=None, CacheParameterGroupName='default.redis6.0.clus
    """Creates an ElastiCache Cluster with cluster mode enabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
    cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
    used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximum 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
    replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster.
    :param CacheParameterGroupName: Parameter group to be used. Must be compatible with
    the engine version and cluster mode enabled.
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'
    elif not isinstance(UserGroupIds, (list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'redis',
```

```
        'EngineVersion': EngineVersion,
        'ReplicationGroupDescription': ReplicationGroupDescription,
        'ReplicationGroupId': ReplicationGroupId,
        'SnapshotRetentionLimit': 30,
        'TransitEncryptionEnabled': True,
        'UserGroupIds': UserGroupIds,
        'NumNodeGroups': NumNodeGroups,
        'ReplicasPerNodeGroup': ReplicasPerNodeGroup,
        'CacheParameterGroupName': CacheParameterGroupName
    }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds, (list)):
        params.update({'SecurityGroupIds': SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName': CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':
    # Creates a cluster mode enabled cluster
    response = create_cluster_mode_enabled(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        ReplicationGroupDescription='Redis cluster mode enabled with replicas',
        ReplicationGroupId='redis2021',
        # Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
        # (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=2,
        ReplicasPerNodeGroup=1,
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
        CacheSubnetGroupName='default'

    )

    logging.info(response)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ClusterModeEnabledWithRBAC.py
```

詳細については、「[クラスターの管理](#)」を参照してください。

ユーザー/ユーザーグループが存在するかどうかを確認し、そうでない場合は作成する

RBAC では、ユーザーを作成し、アクセス文字列を使用して特定のアクセス許可を割り当てます。ユーザーを特定の役割 (管理者、人事) に割り当てたユーザーグループに割り当て、そのユーザーグループを 1 ElastiCache つ以上の Redis レプリケーショングループにデプロイします。これにより、同じ Redis レプリケーショングループを使用するクライアント間にセキュリティ境界を設定し、クライアントが互いのデータにアクセスできないようにすることができます。詳細については、「[ロールベースのアクセスコントロール \(RBAC\)](#)」を参照してください。

次のプログラムをコピーして、.py UserAndUserGroups という名前のファイルに貼り付けます。認証情報を提供するメカニズムを更新します。この例の認証情報は交換可能と表示され、宣言されていない項目が割り当てられています。認証情報をハードコーディングすることは避けてください。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def check_user_exists(UserId):
    """Checks if UserId exists

    Returns True if UserId exists, otherwise False
    :param UserId: ElastiCache User ID
    :return: True|False
    """
    try:
        response = client.describe_users(
            UserId=UserId,
        )
        if response['Users'][0]['UserId'].lower() == UserId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserNotFound':
            logging.info(e.response['Error'])
            return False
        else:
```

```
        raise

def check_group_exists(UserGroupId):
    """Checks if UserGroupID exists

    Returns True if Group ID exists, otherwise False
    :param UserGroupId: ElastiCache User ID
    :return: True|False
    """

    try:
        response = client.describe_user_groups(
            UserGroupId=UserGroupId
        )
        if response['UserGroups'][0]['UserGroupId'].lower() == UserGroupId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserGroupNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def create_user(UserId=None, Username=None, Password=None, AccessString=None):
    """Creates a new user

    Returns the ARN for the newly created user or the error message
    :param UserId: ElastiCache user ID. User IDs must be unique
    :param Username: ElastiCache user name. ElastiCache allows multiple users with the
    same name as long as the associated user ID is unique.
    :param Password: Password for user. Must have at least 16 chars.
    :param AccessString: Access string with the permissions for the user. For
    details refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Clusters.RBAC.html#Access-string
    :return: user ARN
    """

    try:
        response = client.create_user(
            UserId=UserId,
            Username=Username,
            Engine='Redis',
            Passwords=[Password],
            AccessString=AccessString,
            NoPasswordRequired=False
        )
```

```

    )
    return response['ARN']
except Exception as e:
    logging.info(e.response['Error'])
    return e.response['Error']

def create_group(UserGroupId=None, UserIds=None):
    """Creates a new group.
    A default user is required (mandatory) and should be specified in the UserIds list

    Return: Group ARN
    :param UserIds: List with user IDs to be associated with the new group. A default
    user is required
    :param UserGroupId: The ID (name) for the group
    :return: Group ARN
    """
    try:
        response = client.create_user_group(
            UserGroupId=UserGroupId,
            Engine='Redis',
            UserIds=UserIds
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])

if __name__ == '__main__':

    groupName='mygroup2'
    userName = 'myuser2'
    userId=groupName+'-'+userName

    # Creates a new user if the user ID does not exist.
    for tmpUserId,tmpUserName in [ (userId,userName), (groupName+'-
default','default')]:
        if not check_user_exists(tmpUserId):
            response=create_user(UserId=tmpUserId,
UserName=EXAMPLE,Password=EXAMPLE,AccessString='on ~* +@all')
            logging.info(response)
            # assigns the new user ID to the user group
        if not check_group_exists(groupName):
            UserIds = [ userId , groupName+'-default']
            response=create_group(UserGroupId=groupName,UserIds=UserIds)

```

```
logging.info(response)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python UserAndUserGroups.py
```

ElastiCache への接続

次の例では、Redis クライアントを使用して ElastiCache に接続します。

トピック

- [クラスタモードが無効のクラスターへの接続](#)
- [クラスタモードが有効のクラスターへの接続](#)

クラスタモードが無効のクラスターへの接続

次のプログラムを `ConnectClusterModeDisabled.py` というファイルにコピーアンドペーストします。認証情報を提供するメカニズムを更新します。この例の認証情報は交換可能と表示され、宣言されていない項目が割り当てられています。認証情報をハードコーディングすることは避けてください。

```
from redis import Redis
import logging

logging.basicConfig(level=logging.INFO)
redis = Redis(host='primary.xxx.yyyyyy.zzz1.cache.amazonaws.com', port=6379,
              decode_responses=True, ssl=True, username=example, password=EXAMPLE)

if redis.ping():
    logging.info("Connected to Redis")
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ConnectClusterModeDisabled.py
```

クラスタモードが有効のクラスターへの接続

次のプログラムを `ConnectClusterModeEnabled.py` というファイルにコピーアンドペーストします。

```
from rediscluster import RedisCluster
import logging

logging.basicConfig(level=logging.INFO)
redis = RedisCluster(startup_nodes=[{"host":
    "xxx.yyy.clustercfg.zzz1.cache.amazonaws.com", "port": "6379"}],
    decode_responses=True, skip_full_coverage_check=True)

if redis.ping():
    logging.info("Connected to Redis")
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ConnectClusterModeEnabled.py
```

使用例

次の例では、ElastiCache 用の boto3 SDK を使用して ElastiCache を使用します。

トピック

- [文字列の設定と取得](#)
- [複数の項目があるハッシュを設定して取得する](#)
- [Pub/Sub チャンネルから公開 \(書き込み\) および登録 \(読み取り\) する](#)
- [ストリームからの書き込みおよび読み取り](#)

文字列の設定と取得

次のプログラムを SetAndGetStrings.py というファイルにコピーアンドペーストします。

```
import time
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s: %(message)s')

keyName='mykey'
currTime=time.ctime(time.time())

# Set the key 'mykey' with the current date and time as value.
# The Key will expire and removed from cache in 60 seconds.
redis.set(keyName, currTime, ex=60)
```



```
# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieve the key value and current TTL
keyValue=redis.get(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
    keyValue, keyTTL))
```

このプログラムを実行するには、次のコマンドを入力します。

```
python SetAndGetStrings.py
```

複数の項目があるハッシュを設定して取得する

次のプログラムを SetAndGetHash.py というファイルにコピーアンドペーストします。

```
import logging
import time

logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
keyValues={'datetime': time.ctime(time.time()), 'epochtime': time.time()}

# Set the hash 'mykey' with the current date and time in human readable format
# (datetime field) and epoch number (epochtime field).
redis.hset(keyName, mapping=keyValues)

# Set the key to expire and removed from cache in 60 seconds.
redis.expire(keyName, 60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieves all the fields and current TTL
keyValues=redis.hgetall(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
    keyValues, keyTTL))
```

このプログラムを実行するには、次のコマンドを入力します。

```
python SetAndGetHash.py
```

Pub/Sub チャンネルから公開 (書き込み) および登録 (読み取り) する

次のプログラムを PubAndSub.py というファイルにコピーアンドペーストします。

```
import logging
import time

def handlerFunction(message):
    """Prints message got from PubSub channel to the log output

    Return None
    :param message: message to log
    """
    logging.info(message)

logging.basicConfig(level=logging.INFO)
redis = Redis(host="redis202104053.tihewd.ng.0001.use1.cache.amazonaws.com", port=6379,
              decode_responses=True)

# Creates the subscriber connection on "mychannel"
subscriber = redis.psubsub()
subscriber.subscribe(**{'mychannel': handlerFunction})

# Creates a new thread to watch for messages while the main process continues with its
# routines
thread = subscriber.run_in_thread(sleep_time=0.01)

# Creates publisher connection on "mychannel"
redis.publish('mychannel', 'My message')

# Publishes several messages. Subscriber thread will read and print on log.
while True:
    redis.publish('mychannel',time.ctime(time.time()))
    time.sleep(1)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python PubAndSub.py
```

ストリームからの書き込みおよび読み取り

次のプログラムを `ReadWriteStream.py` というファイルにコピーアンドペーストします。

```
from redis import Redis
import redis.exceptions as exceptions
import logging
import time
import threading

logging.basicConfig(level=logging.INFO)

def writeMessage(streamName):
    """Starts a loop writting the current time and thread name to 'streamName'

    :param streamName: Stream (key) name to write messages.
    """
    fieldsDict={'writerId':threading.currentThread().getName(),'myvalue':None}
    while True:
        fieldsDict['myvalue'] = time.ctime(time.time())
        redis.xadd(streamName,fieldsDict)
        time.sleep(1)

def readMessage(groupName=None,streamName=None):
    """Starts a loop reading from 'streamName'
    Multiple threads will read from the same stream consumer group. Consumer group is
    used to coordinate data distribution.
    Once a thread acknowleges the message, it won't be provided again. If message
    wasn't acknowledged, it can be served to another thread.

    :param groupName: stream group were multiple threads will read.
    :param streamName: Stream (key) name where messages will be read.
    """

    readerID=threading.currentThread().getName()
    while True:
        try:
            # Check if the stream has any message
            if redis.xlen(streamName)>0:
                # Check if if the messages are new (not acknowledged) or not (already
                processed)
                streamData=redis.xreadgroup(groupName,readerID,
                {streamName:'>'},count=1)
                if len(streamData) > 0:
```

```
        msgId,message = streamData[0][1][0]
        logging.info("{}: Got {} from ID
{}".format(readerID,message,msgId))
        #Do some processing here. If the message has been processed
sucessfully, acknowledge it and (optional) delete the message.
        redis.xack(streamName,groupName,msgId)
        logging.info("Stream message ID {} read and processed successfully
by {}".format(msgId,readerID))
        redis.xdel(streamName,msgId)
    else:
        pass
except:
    raise

    time.sleep(0.5)

# Creates the stream 'mystream' and consumer group 'myworkergroup' where multiple
threads will write/read.
try:
    redis.xgroup_create('mystream','myworkergroup',mkstream=True)
except exceptions.ResponseError as e:
    logging.info("Consumer group already exists. Will continue despite the error:
{}".format(e))
except:
    raise

# Starts 5 writer threads.
for writer_no in range(5):
    writerThread = threading.Thread(target=writeMessage, name='writer-'+str(writer_no),
args=('mystream',),daemon=True)
    writerThread.start()

# Starts 10 reader threads
for reader_no in range(10):
    readerThread = threading.Thread(target=readMessage, name='reader-'+str(reader_no),
args=('myworkergroup','mystream',),daemon=True)
    readerThread.daemon = True
    readerThread.start()

# Keep the code running for 30 seconds
time.sleep(30)
```

このプログラムを実行するには、次のコマンドを入力します。

```
python ReadWriteStream.py
```

チュートリアル:Amazon VPC ElastiCache 内のアマゾンにアクセスするための Lambda 関数の設定

このチュートリアルでは、ElastiCache サーバーレスキャッシュを作成する方法、Lambda 関数を作成する方法、Lambda 関数をテストする方法、およびオプションでその後にクリーンアップする方法を学ぶことができます。

トピック

- [ステップ 1: サーバーレスキャッシュを作成する](#)
- [ステップ 2: Lambda 関数を作成する](#)
- [ステップ 3: Lambda 関数をテストする](#)
- [ステップ 4: クリーンアップ \(オプション\)](#)

ステップ 1: サーバーレスキャッシュを作成する

サーバーレスキャッシュを作成するには、次の手順に従います。

トピック

- [ステップ 1.1: サーバーレスキャッシュを作成する](#)
- [ステップ 1.2: サーバーレスキャッシュエンドポイントをコピーする](#)
- [ステップ 1.3: IAM ロールを作成する](#)
- [ステップ 1.4: サーバーレスキャッシュを作成する](#)

ステップ 1.1: サーバーレスキャッシュを作成する

このステップでは、(CLI) を使用して、アカウントの us-east-1 リージョンのデフォルトの Amazon VPC にサーバーレスキャッシュを作成します。AWS Command Line Interface コンソールまたは API を使用してサーバーレスキャッシュを作成する方法については、ElastiCache を参照してください。[ステップ 1: キャッシュを作成する](#)

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name cache-01 \
```

```
--description "ElastiCache IAM auth application" \  
--engine redis
```

[ステータス] フィールドの値が `CREATING` に設定されていることに注意してください。ElastiCache キャッシュの作成が完了するまでに 1 分かかることがあります。

ステップ 1.2: サーバーレスキャッシュエンドポイントをコピーする

ElastiCache for Redis がコマンドでキャッシュの作成を完了したことを確認します。describe-serverless-caches

```
aws elasticache describe-serverless-caches \  
--serverless-cache-name cache-01
```

出力に表示されたエンドポイントアドレスをコピーします。Lambda 関数のデプロイパッケージを作成するときに、このアドレスが必要になります。

ステップ 1.3: IAM ロールを作成する

1. アカウントが新しいロールを引き継ぐことを許可するロール用の IAM 信頼ポリシードキュメントを以下に示すように作成します。ポリシーを trust-policy.json というファイルに保存します。

```
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },  
    "Action": "sts:AssumeRole"  
  },  
  {  
    "Effect": "Allow",  
    "Principal": {  
      "Service": "lambda.amazonaws.com"  
    },  
    "Action": "sts:AssumeRole"  
  }  
]
```

2. 以下に示すように、IAM ポリシードキュメントを作成します。ポリシーを policy.json というファイルに保存します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

3. IAM ロールを作成します。

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. IAM ポリシーを作成します。

```
aws iam create-policy \
--policy-name "elasticache-allow-all" \
--policy-document file://policy.json
```

5. IAM ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
--role-name "elasticache-iam-auth-app" \
--policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

ステップ 1.4: サーバーレスキャッシュを作成する

1. 新しいデフォルトユーザーを作成します。

```
aws elasticache create-user \
--user-name default \
```

```
--user-id default-user-disabled \  
--engine redis \  
--authentication-mode Type=no-password-required \  
--access-string "off +get ~keys*"
```

2. IAM を有効にしている新しいユーザーを作成します。

```
aws elasticache create-user \  
  --user-name iam-user-01 \  
  --user-id iam-user-01 \  
  --authentication-mode Type=iam \  
  --engine redis \  
  --access-string "on ~* +@all"
```

3. ユーザーグループを作成し、ユーザーをアタッチします。

```
aws elasticache create-user-group \  
  --user-group-id iam-user-group-01 \  
  --engine redis \  
  --user-ids default-user-disabled iam-user-01  
  
aws elasticache modify-serverless-cache \  
  --serverless-cache-name cache-01 \  
  --user-group-id iam-user-group-01
```

ステップ 2: Lambda 関数を作成する

Lambda 関数を作成するには、次の手順を実行します。

トピック

- [ステップ 2.1: Lambda 関数を作成する](#)
- [ステップ 2.2: IAM ロール \(実行ロール\) を作成する](#)
- [ステップ 2.3: デプロイパッケージをアップロードする \(Lambda 関数を作成する\)](#)

ステップ 2.1: Lambda 関数を作成する

このチュートリアルでは、Lambda 関数用の Python のサンプルコードを提供します。

Python

次の Python コードの例は、ElastiCache キャッシュに項目を読み書きします。コードを `app.py` という名前のファイルに保存します。 `elasticache_endpoint` コード内の値は必ず、ステップ 1.2 でコピーしたエンドポイントアドレスに置き換えてください。

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import boto3.session
import redis
from boto3.model import ServiceId
from boto3.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class ElastiCacheIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cache_name, is_serverless=False, region="us-east-1"):
        self.user = user
        self.cache_name = cache_name
        self.is_serverless = is_serverless
        self.region = region

        session = boto3.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("elasticache"),
            self.region,
            "elasticache",
            "v4",
            session.get_credentials(),
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}
        if self.is_serverless:
            query_params["ResourceType"] = "ServerlessCache"
        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cache_name,
                path="/",
                query=urlencode(query_params),
```

```
        params="",
        fragment="",
    )
)
signed_url = self.request_signer.generate_presigned_url(
    {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
    operation_name="connect",
    expires_in=900,
    region_name=self.region,
)
# RequestSigner only seems to work if the URL has a protocol, but
# Elasticache only accepts the URL without a protocol
# So strip it off the signed URL before returning
return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cache_name = "cache-01" # replace with your cache name
    elasticache_endpoint = "cache-01-xxxxx.serverless.us-east-1.cache.amazonaws.com" #
    # replace with your cache endpoint
    creds_provider = ElastiCacheIAMProvider(user=username, cache_name=cache_name,
    is_serverless=True)
    redis_client = redis.Redis(host=elasticache_endpoint, port=6379,
    credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
    # check the retrieved item matches the item added to the cache and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from Redis.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
        {decoded_result}")

    return "Fetched value from Redis"
```

このコードは Python redis-py ライブラリを使用してアイテムをキャッシュに格納し、取得します。このコードは cachetools を使用して、生成された IAM Auth トークンを 15 分間キャッシュします。redis-py と cachetools を含むデプロイパッケージを作成するには、以下の手順を実行します。

app.py ソースコードファイルを含むプロジェクトディレクトリに、redis-py と cachetools ライブラリをインストールするフォルダパッケージを作成します。

```
mkdir package
```

pip を使用して redis-py とキャッシュツールをインストールします。

```
pip install --target ./package redis
pip install --target ./package cachetools
```

redis-py ライブラリと cachetools ライブラリを含む.zip ファイルを作成します。Linux および macOS では、次のコマンドを実行します。Windows では、お好みの zip ユーティリティを使用して、redis-py ライブラリと cachetools ライブラリをルートにした.zip ファイルを作成します。

```
cd package
zip -r ../my_deployment_package.zip .
```

.zip ファイルに関数コードを追加します。Linux および macOS では、次のコマンドを実行します。Windows では、お好みの zip ユーティリティを使用して app.py を.zip ファイルのルートに追加します。

```
cd ..
zip my_deployment_package.zip app.py
```

ステップ 2.2: IAM ロール (実行ロール) を作成する

AWS AWSLambdaVPCAccessExecutionRole という名前の管理ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
  --role-name "elasticache-iam-auth-app" \
  --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

ステップ 2.3: デプロイパッケージをアップロードする (Lambda 関数を作成する)

このステップでは、AWS CLI `create-function` コマンドを使用して Lambda 関数 (`AccessRedis`) を作成します。

デプロイパッケージの `.zip` ファイルを含むプロジェクトディレクトリから、次の Lambda CLI コマンドを実行します。 `create-function`

ロールオプションには、ステップ 2.2 で作成した実行ロールの ARN を使用します。 `vpc-config` には、デフォルト VPC のサブネットとデフォルト VPC のセキュリティグループ ID をカンマで区切って入力します。これらの値は Amazon VPC コンソールにあります。デフォルト VPC のサブネットを見つけるには、`[Your VPC]` を選択し、AWS 次にアカウントのデフォルト VPC を選択します。この VPC のセキュリティグループを見つけるには、`[セキュリティ]` に移動し、`[セキュリティグループ]` を選択します。 `us-east-1` リージョンが選択されていることを確認します。

```
aws lambda create-function \  
--function-name AccessRedis \  
--region us-east-1 \  
--zip-file fileb://my_deployment_package.zip \  
--role arn:aws:iam::123456789012:role/elasticache-iam-auth-app \  
--handler app.lambda_handler \  
--runtime python3.12 \  
--timeout 30 \  
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id
```

ステップ 3: Lambda 関数をテストする

このステップでは、`invoke` コマンドを使用して Lambda 関数を手動で呼び出します。Lambda 関数が実行されると、UUID が生成され、Lambda ElastiCache コードで指定したキャッシュに書き込みます。次に、Lambda 関数はキャッシュから項目を取得します。

1. `invoke` コマンドを使用して Lambda 関数 (`AccessRedis`) を呼び出します。AWS Lambda

```
aws lambda invoke \  
--function-name AccessRedis \  
--region us-east-1 \  
output.txt
```

2. Lambda 関数が正常に実行されたことを、次のように確認します。

- output.txt ファイルを確認します。
- CloudWatch コンソールを開き、CloudWatch 関数のロググループ (/aws/lambda/) を選択して、Logs の結果を確認します。AccessRedisログストリームには、以下と同様のコマンドの出力が含まれます。

```
Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
826e70c5f4d2478c8c18027125a3e01e from Redis.
```

- コンソールで結果を確認します。AWS Lambda

ステップ 4: クリーンアップ (オプション)

クリーンアップするには、次の手順を実行します。

トピック

- [ステップ 4.1: Lambda 関数を削除する](#)
- [ステップ 4.2: サーバーレスキャッシュを削除する](#)
- [ステップ 4.3: IAM ロールとポリシーを削除する](#)

ステップ 4.1: Lambda 関数を削除する

```
aws lambda delete-function \  
--function-name AccessRedis
```

ステップ 4.2: サーバーレスキャッシュを削除する

キャッシュを削除する。

```
aws elasticache delete-serverless-cache \  
--serverless-cache-name cache-01
```

ユーザーとユーザーグループを削除します。

```
aws elasticache delete-user \  
--user-id default-user-disabled  
  
aws elasticache delete-user \  

```

```
--user-id iam-user-01
```

```
aws elasticache delete-user-group \  
--user-group-id iam-user-group-01
```

ステップ 4.3: IAM ロールとポリシーを削除する

```
aws iam detach-role-policy \  
--role-name "elasticache-iam-auth-app" \  
--policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

```
aws iam detach-role-policy \  
--role-name "elasticache-iam-auth-app" \  
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

```
aws iam delete-role \  
--role-name "elasticache-iam-auth-app"
```

```
aws iam delete-policy \  
--policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

独自の ElastiCache クラスターの設計と管理

ElastiCache クラスターをきめ細かく制御する必要がある場合は、独自のクラスターを設計することができます。ElastiCache では、ノードベースのクラスターを運用できます。その場合は、クラスターのノードタイプ、ノード数、AWS アベイラビリティーゾーン間のノード配置を選択します。ElastiCache はフルマネージド型のサービスであるため、クラスターのハードウェアプロビジョニング、モニタリング、ノード交換、ソフトウェアのパッチ適用は自動管理されます。

セットアップに関する詳細については、「[セットアップ](#)」を参照してください。ノードまたはクラスターの管理、更新、削除の詳細については、「[ノードの管理](#)」を参照してください。独自の ElastiCache クラスターを設計する場合の Amazon ElastiCache のデプロイの主要コンポーネントの概要については、以下の[重要な概念](#)を参照してください。

トピック

- [ElastiCache Redis のコンポーネントと機能用](#)
- [ElastiCache for Redis の用語](#)
- [独自のクラスターの設計](#)
- [ノードの管理](#)
- [クラスターの管理](#)
- [Memcached キャッシュと Redis の独自設計型キャッシュの比較](#)
- [ElastiCache へのオンライン移行](#)
- [リージョンとアベイラビリティーゾーンの選択](#)

ElastiCache Redis のコンポーネントと機能用

以下に、Amazon ElastiCache デプロイメントの主要コンポーネントの概要を示します。

トピック

- [ElastiCache ノード](#)
- [ElastiCache Redis シャード用](#)
- [ElastiCache Redis クラスター用](#)
- [ElastiCache Redis レプリケーション用](#)
- [AWS リージョンとアベイラビリティーゾーン](#)

- [ElastiCache Redis エンドポイント用](#)
- [ElastiCache パラメータグループ](#)
- [ElastiCache Redis のセキュリティ用](#)
- [ElastiCache サブネットグループ](#)
- [ElastiCache Redis バックアップ用](#)
- [ElastiCache イベント](#)

ElastiCache ノード

ElastiCache ノードはデプロイメントの最小構成要素です。ノードは他のノードから分離するか、一定の関係を設定できます。

ノードは、安全なネットワークに接続された RAM の固定サイズの断片です。各ノードは、クラスター作成時に選択したエンジンとバージョンのインスタンスを実行します。必要に応じて、異なるインスタンスタイプにノードのクラスターを拡大または縮小できます。詳細については、「[Redis ElastiCache のスケーリング](#)」を参照してください。

クラスター内の各ノードは同じインスタンスタイプで、同じキャッシュエンジンを実行します。各キャッシュノードはそれぞれ Domain Name Service (DNS) 名とポートを持っています。それぞれ関連付けられている異なるメモリ量で、複数のタイプのキャッシュノードがサポートされています。サポートされるインスタンスタイプノードのリストについては、「[サポートされているノードの種類](#)」を参照してください。

pay-as-you-go ノードは基本購入が可能で、お支払いいただくのはノードの使用分のみです。または、大幅な割引が適用される時間単価制でリザーブドノードを購入することもできます。使用率が高い場合は、リザーブドノードを購入するほうがコストを削減できます。クラスターを常に使用しており、急激な使用率の増加には一時的にノードを追加して対処しているとします。この場合、多くのリザーブドノードを購入して、ほとんど常時実行できます。その後、pay-as-you-go ときどきノードを追加する必要が生じた場合に備えて、ノードを購入できます。リザーブドノードの詳細については、「[ElastiCache のリザーブドノード](#)」を参照してください。

ノードの詳細については、「[ノードの管理](#)」を参照してください。

ElastiCache Redis シャード用

Redis シャード (API および CLI ではノードグループと呼ばれる) は、1~6 の関連ノードのグループです。Redis (クラスターモードが無効) クラスターには常に少なくとも 1 つのシャードがあります。

シャーディングは、大規模なデータベースを、データシャードと呼ばれる、より小さく、より速く、管理しやすい部分に分割するデータベースを分割する方法です。これにより、操作を複数のセクションに分散させることができるため、データベースの効率が向上します。シャードを使用すると、パフォーマンス、スケーラビリティ、コスト効率の向上など、多くのメリットが得られます。

Redis (クラスターモード有効) クラスターには、最大 500 個のシャードがあり、データはシャード間で分割されます。Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

[複数ノードシャード] では、1 つの読み書き可能プライマリノードと 1~5 個のレプリカノードを含めることで、レプリケーションを実装します。詳細については、「[レプリケーショングループを使用する高可用性](#)」を参照してください。

シャードの詳細については、「[シャードの使用](#)」を参照してください。

ElastiCache Redis クラスター用

Redis クラスター は、単一または複数の [ElastiCache Redis シャード用](#) の論理グループです。データは、Redis (クラスターモードが有効) クラスター内のシャード間で分割されます。

ElastiCache 多くの操作はクラスターを対象としています。

- クラスターの作成
- クラスターの変更
- クラスター (Redis のすべてのバージョン) のスナップショットを作成する
- クラスターの削除
- クラスターのエレメントの表示
- クラスター間で送受信されるコスト配分タグの追加または削除

詳細については、次の関連トピックを参照してください。

- [クラスタの管理](#) および [ノードの管理](#)

クラスタ、ノードおよび関連オペレーションに関する情報。

- [AWS サービス制限:Amazon ElastiCache](#)

ElastiCache ノードやクラスタの最大数などの制限に関する情報。これらの制限の一部を超える場合は、[Amazon ElastiCache キャッシュノードリクエストフォームを使用してリクエストを行うことができます](#)。

- [障害の軽減](#)

クラスタおよびレプリケーショングループの耐障害性向上に関する情報。

一般的なクラスタの設定

以下は一般的なクラスタの構成です。

Redis クラスタ

Redis (クラスタモードが無効) クラスタには、常に 1 個のシャード (API および CLI では 1 つのノードグループ) のみが含まれます。Redis シャードには、1~6 個のノードが含まれます。シャードに複数のノードがある場合、シャードはレプリケーションをサポートします。この場合、1 つのノードは読み取り/書き込みプライマリノードであり、他のノードは読み取り専用レプリカノードです。

耐障害性を高めるために、Redis クラスタに 2 つ以上のノードを含め、マルチ AZ を有効にすることをお勧めします。詳細については、「[障害の軽減](#)」を参照してください。

Redis (クラスタモードが無効) クラスタの需要が変化した場合は、スケールアップまたはスケールダウンできます。そのためには、クラスタを別のノードインスタンスタイプに移動します。読み取り負荷の高いアプリケーションの場合、読み取り専用レプリカ Redis (クラスタモードが無効) クラスタを追加することをお勧めします。これにより、読み取りをより適切な数のノードに分散させることができます。

データ階層化を使用することもできます。アクセス頻度の高いデータはメモリに保存され、アクセス頻度の低いデータはディスクに保存されます。データ階層化を使用する上での利点は、必要なメモリ容量を削減できることです。詳細については、「[データ階層化](#)」を参照してください。

ElastiCache Redis (クラスタモードが無効) クラスタのノードタイプをより大きなノードタイプに動的に変更することをサポートします。スケールアップ/ダウンの詳細については、「[Redis \(ク](#)

[ラスターモードが無効\)の単一ノードクラスターのスケールリング](#)」または「[レプリカノードを含む Redis \(クラスターモードが無効\) クラスターのスケールリング](#)」を参照してください。

ElastiCache Redis レプリケーション用

レプリケーションは、2~6個のノードをシャード (API と CLI ではノードグループと呼ばれる) にまとめることで実装されます。これらのノードの1つは読み書き可能プライマリノードです。他のすべてのノードは読み取り専用レプリカノードです。

各レプリカノードは、プライマリノードからのデータのコピーを維持します。レプリカノードは、非同期レプリケーションメカニズムを使用して、プライマリノードとの同期を維持します。アプリケーションは、クラスターのどのノードからでも読み込みことができますが、書き込むことができるのはプライマリノードのみになります。リードレプリカは、読み取りを複数のエンドポイントに分散させることで拡張できます。リードレプリカは、データの複数のコピーを維持することで、耐障害性が向上します。複数のアベイラビリティゾーンにリードレプリカを配置することで、耐障害性が向上します。耐障害性の詳細については、「[障害の軽減](#)」を参照してください。

Redis (クラスターモードが無効) クラスターでは、1個のシャード (API および CLI では、ノードグループ) をサポートします

API と CLI の観点からのレプリケーションでは、以前のバージョンとの互換性を維持するために異なる用語を使用していますが、結果は同じです。以下の表では、レプリケーションの実装に関する API および CLI の用語を示しています。

レプリケーションの比較: Redis (クラスターモードが無効) および Redis (クラスターモードが有効)

次の表には、Redis (クラスターモードが無効) と Redis (クラスターモードが有効) レプリケーショングループの各機能の比較があります。

	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
シャード (ノードグループ)	1	1~500
各シャードあたりのレプリカ数 (ノードグループ)	0~5	0~5
データのパーティション化	No	Yes
レプリカの追加/削除	はい	Yes

	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
ノードグループの追加/削除	No	Yes
サポートの拡大	はい	Yes
エンジンアップグレードのサポート	はい	Yes
レプリカをプライマリーに昇格	Yes	自動
マルチ AZ	オプションです。	必須
バックアップ/復元	はい	Yes

注意:

どのプライマリーにもレプリカがなく、プライマリーに障害が発生した場合、そのプライマリーのデータがすべて失われます。

バックアップと復元を使用して Redis に移行できます (クラスターモードが有効)。

バックアップと復元を使用して Redis (クラスターモードが有効) クラスターのサイズを変更できます。

すべてのシャード (API と CLI ではノードグループ) とノードは同じ AWS リージョンに存在する必要があります。ただし、AWS そのリージョン内の複数のアベイラビリティゾーンに個々のノードをプロビジョニングすることはできます。

リードレプリカは、データが 2 つ以上のノード (プライマリーと 1 つ以上のリードレプリカ) でレプリケートされるため、潜在的なデータ損失から保護します。信頼性を高め、より迅速な復旧を可能にするには、異なるアベイラビリティゾーンに 1 つ以上のリードレプリカを作成することをお勧めします。

グローバルデータストアを利用することもできます。Global Datastore for Redis 機能を使用すると、AWS フルマネージド型の高速で信頼性が高く安全なレプリケーションをリージョン間で行うことができます。この機能を使用すると、for Redis ElastiCache 用のクロスリージョンリードレプリカクラスターを作成して、リージョンをまたがる低レイテンシーの読み取りとディザスタリカバリが可能

能になります。AWS 詳細については、「[AWS グローバルデータストアを使用したリージョン間のレプリケーション](#)」を参照してください。

レプリケーション: 制限と例外

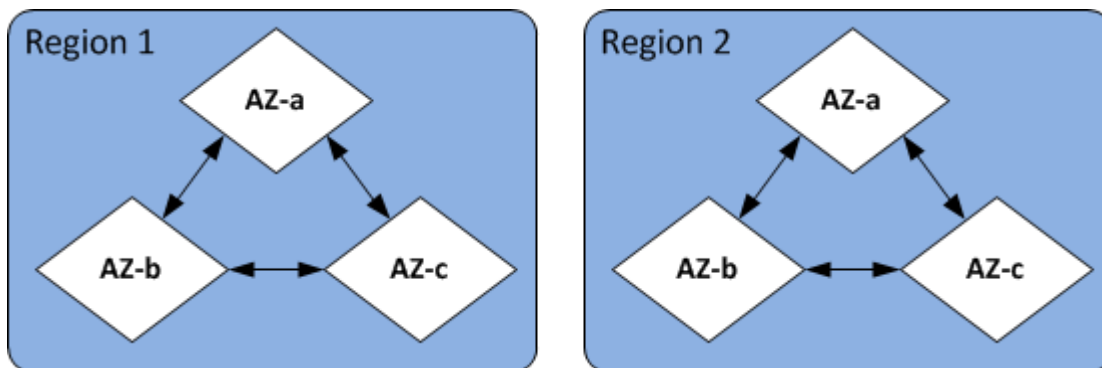
- マルチ AZ は、T1 ノードタイプではサポートされません。

AWS リージョンとアベイラビリティーゾーン

Amazon ElastiCache AWS は世界中の複数の地域でご利用いただけます。そのため、ElastiCache ビジネス要件を満たす場所でクラスターを起動できます。たとえば、AWS 顧客に最も近いリージョンや、特定の法的要件を満たすリージョンで起動できます。

デフォルトでは、AWS SDK、ElastiCache API AWS CLI、ElastiCache コンソールは米国西部 (オレゴン) リージョンを参照します。ElastiCache AWS 新しいリージョンへの提供が拡大するにつれ、AWS これらのリージョンの新しいエンドポイントも利用できるようになります。これらは HTTP リクエスト、AWS SDK AWS CLI、およびコンソールで使用できます。ElastiCache

AWS AWS 各リージョンは他の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。AWS リージョンとアベイラビリティーゾーンの詳細については、「[リージョンとアベイラビリティーゾーンの選択](#)」を参照してください。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を大まかに示しています。



AWS ElastiCache がサポートするリージョンとそのエンドポイントについては、「[サポートされているリージョンおよびエンドポイント](#)」を参照してください。

ElastiCache Redis エンドポイント用

エンドポイントは、ElastiCache アプリケーションがノードまたはクラスターに接続するために使用する固有のアドレスです。

Redis (クラスターモードが無効) の単一ノードエンドポイント

単一ノード Redis クラスターのエンドポイントは、読み取りおよび書き込みのためにクラスターに接続するのに使用されます。

Redis (クラスターモードが無効) のマルチノードエンドポイント

マルチノード Redis (クラスターモードが無効) クラスターには、2 種類のエンドポイントがあります。プライマリエンドポイントは常に、プライマリロールで特定のノードが変わっても、クラスター内のプライマリノードに接続します。クラスターへのすべての書き込みには、プライマリエンドポイントを使用します。

読み込みエンドポイントを使用して、すべてのリードレプリカ間でエンドポイントへの着信接続を均等に分割します。個々のノードエンドポイント (API/CLI ではリードエンドポイント) を読み取りオペレーションに使用します。

Redis (クラスターモードが有効) エンドポイント

Redis (クラスターモードが有効) クラスターには、単一の設定エンドポイントがあります。設定エンドポイントに接続することで、アプリケーションはクラスター内のシャードごとにプライマリおよびリードエンドポイントを検出できます。

詳細については、「[接続エンドポイントの検索](#)」を参照してください。

ElastiCache パラメータグループ

キャッシュパラメータグループは、サポートされるエンジンソフトウェアのランタイム設定を管理する簡単な方法です。パラメーターは、メモリの使用状況、削除のポリシー、項目サイズなどを制御するために使用されます。ElastiCache パラメータグループは、クラスターに適用できるエンジン固有のパラメーターの名前付きのコレクションです。これにより、そのクラスター内のすべてのノードがまったく同じ方法で設定されていることを確認します。

サポートされているパラメータのリスト、デフォルト値、変更可能なパラメーターについては、「[DescribeEngineDefaultParameters](#)」 (CLI: [describe-engine-default-parameters](#)) を参照してください。

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

ElastiCache Redis のセキュリティ用

ElastiCache セキュリティ強化のため、Redis ノードへのアクセスは、許可した Amazon EC2 インスタンスで実行されているアプリケーションに制限されます。セキュリティグループを使用して、クラスターへのアクセスが許可される Amazon EC2 インスタンスを制御できます。

デフォルトでは、Redis ElastiCache クラスターの新しいものはすべて Amazon Virtual Private Cloud (Amazon VPC) 環境で起動されます。サブネットグループを使用して、特定のサブネットで行われている Amazon EC2 インスタンスからのクラスターアクセスを許可できます。

for Redis は、ノードアクセスを制限するだけでなく、ElastiCache 指定されたバージョンの for Redis を実行するノードの TLS とインプレース暗号化をサポートします。ElastiCache 詳細については、次を参照してください。

- [Amazon ElastiCache のデータセキュリティ](#)
- [Redis AUTH コマンドによる認証](#)

ElastiCache サブネットグループ

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

Amazon VPC でクラスターを作成する場合は、キャッシュサブネットグループを指定する必要があります。ElastiCache そのキャッシュサブネットグループを使用して、キャッシュノード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。

Amazon VPC 環境でのキャッシュサブネットグループの使用方法の詳細については、以下を参照してください。

- [Amazon VPC と ElastiCache のセキュリティ](#)
- [ステップ 3: クラスターへのアクセスの許可](#)
- [サブネットおよびサブネットグループ](#)

ElastiCache Redis バックアップ用

バックアップは Redis point-in-time クラスターのコピーです。バックアップは、既存のクラスターを復元するか、または新しいクラスターをシードするのに使用できます。バックアップは、クラスターのすべてのデータといくつかのメタデータで構成されます。

クラスターで実行されている Redis のバージョンによって、バックアッププロセスが成功するためには、異なる予約メモリの量が必要になります。詳細については、次を参照してください。

- [スナップショットおよび復元](#)
- [同期とバックアップの実装方法](#)
- [独自設計型クラスターのバックアップがパフォーマンスに与える影響](#)
- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)

ElastiCache イベント

キャッシュクラスターで重要なイベントが発生すると、特定の Amazon SNS ElastiCache トピックに通知を送信します。これらのイベントとしては、ノードの追加の失敗や成功、セキュリティグループの変更などがあります。主要イベントをモニタリングすることで、クラスターの現在の状態を知り、多くの場合、修正作業を行うことができます。

ElastiCache イベントの詳細については、「」を参照してください [Amazon SNS による ElastiCache イベントのモニタリング](#)。

ElastiCache for Redis の用語

2016 年 10 月に、Amazon ElastiCache は Redis 3.2 のサポートを開始しました。その時点で、データを最大 500 シャード (ElastiCache API と AWS CLI ではノードグループ) に分割するためのサポートを追加しました。以前のバージョンとの互換性を維持するために、新しい Redis の機能を含むように現在の API バージョン 2015-02-02 オペレーションを拡張する予定です。

同時に、この新しい機能に使用される ElastiCache コンソールでは、業界全体で一般的となっている用語を使用する予定です。これらの変化により、どこかの時点で、API および CLI で使用される用語が、コンソールで使用される用語と異なる場合があります。以下のリストでは、API および CLI とコンソールとで異なる場合のある用語を示しています。

キャッシュクラスター/ノードとノードの比較

レプリカノードがない場合、ノードとキャッシュクラスターの間には 1 対 1 の関係があります。したがって、ElastiCache コンソールでは多くの場合、それらの用語を同じ意味で使用していました。現在、コンソールでは一貫してノードという用語を使用しています。唯一の例外は、レプリカノードの有無にかかわらずクラスターの作成プロセスを開始する、[Create Cluster] ボタンです。

ElastiCache API および AWS CLI は以前使用していた用語を使用し続けます。

クラスターとレプリケーショングループの比較

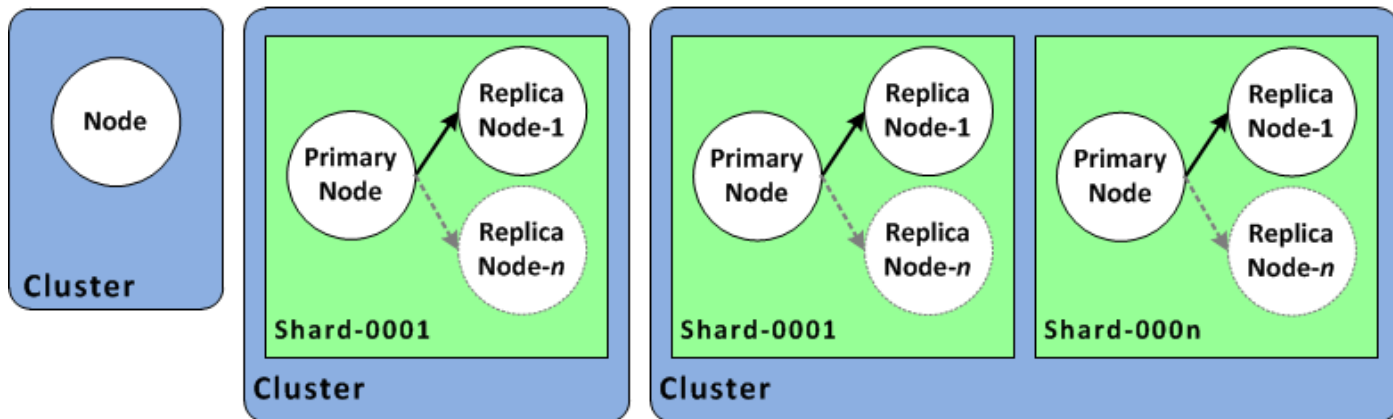
現在、コンソールではクラスターという用語を、すべての ElastiCache for Redis クラスターで使用しています。コンソールはすべての状況下で、クラスターという用語を使用します。

- クラスターが単一ノード Redis クラスターの場合。
- クラスターが、単一のシャード内のレプリケーションをサポートする Redis (クラスターモードが無効) クラスターである場合 (API および CLI では、ノードグループと呼ばれる)。
- クラスターが、1~90 のシャード内のレプリケーションをサポートする Redis (クラスターモードが有効) クラスターの場合、または制限の拡大リクエストで最大 500 のクラスターの場合。この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

レプリケーショングループの詳細については、「[レプリケーショングループを使用する高可用性](#)」を参照してください。

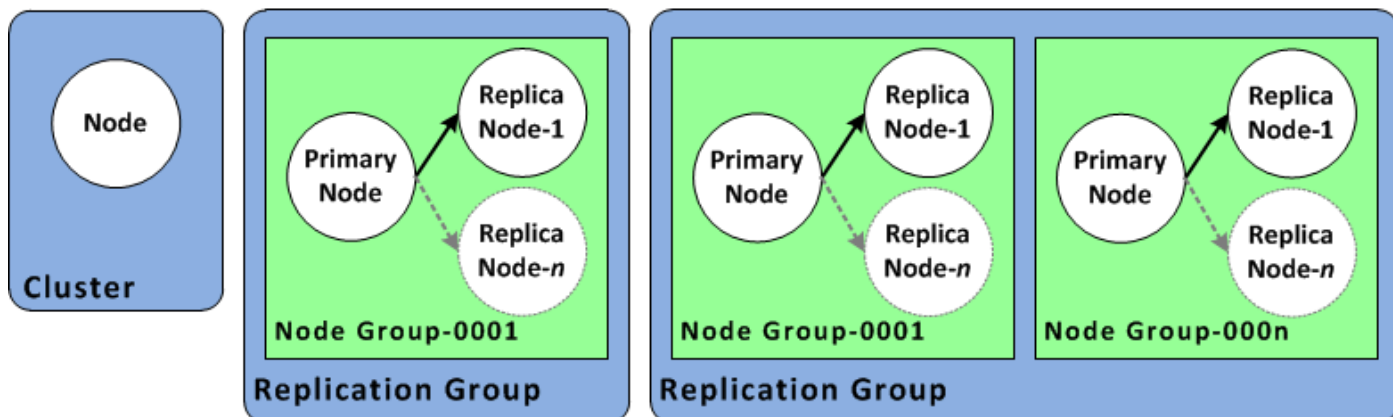
次の図は、コンソールの観点から ElastiCache for Redis クラスターに対する各種トポロジを示しています。

ElastiCache for Redis: Console View



ElastiCache API および AWS CLI オペレーションでは、単一ノードの ElastiCache for Redis クラスターとマルチノードのレプリケーショングループが区別されます。次の図は、ElastiCache API と AWS CLI の観点からさまざまな ElastiCache for Redis トポロジを示したものです。

ElastiCache for Redis: API/CLI View



レプリケーショングループ対グローバルデータストア

グローバルデータストアは、リージョン間で相互にレプリケートする 1 つ以上のクラスターの集合であり、レプリケーショングループは複数のシャードを持つクラスターモードが有効のクラスター間でデータをレプリケートします。Global datastore は、次のもので構成されます。

- [プライマリ (アクティブ) クラスター] – プライマリクラスターは、Global Datastore 内のすべてのクラスターにレプリケートされる書き込みを受け入れます。プライマリクラスターは、読み込みリクエストも受け付けます。
- [セカンダリ (パッシブ) クラスター] – セカンダリクラスターは、読み取りリクエストのみを受け入れ、プライマリクラスターからのデータ更新をレプリケートします。セカンダリクラスターは、プライマリクラスターとは異なる AWS リージョンに存在する必要があります。

グローバルデータストアの詳細については、「[グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)」を参照してください。

独自のクラスターの設計

ElastiCache クラスターの設計を開始するために必要な 1 回限りのアクションは次のとおりです。

トピック

- [セットアップ](#)
- [ステップ 1: サブネットグループの作成](#)
- [ステップ 2: クラスターを作成する](#)
- [ステップ 3: クラスターへのアクセスの許可](#)
- [ステップ 4: クラスターのノードに接続する](#)
- [ステップ 5: クラスターを削除する](#)
- [ElastiCache のチュートリアルと動画](#)
- [次のステップ](#)

セットアップ

クラスターを作成する前に、まずサブネットグループを作成します。キャッシュサブネットグループは、VPC 内でキャッシュクラスターとして指定できるサブネットの集合です。VPC でキャッシュクラスターを起動する場合は、キャッシュサブネットグループを選択する必要があります。次に、ElastiCache ではそのキャッシュサブネットグループを使用して、サブネット内の IP アドレスをクラスター内の各キャッシュノードに割り当てます。

新しいサブネットグループを作成する場合は、使用可能な IP アドレス数に注意してください。サブネットの空き IP アドレス数が非常に少ない場合は、クラスターに追加できるノード数が制約される可能性があります。この問題を解決するために、クラスターのアベイラビリティーゾーンで十分な数の IP アドレスを使用できるように、サブネットグループに 1 つ以上のサブネットを割り当てることができます。その後で、クラスターにノードを追加できます。

ElastiCache のセットアップの詳細については、「[セットアップ](#)」を参照してください。

ステップ 1: サブネットグループの作成

以下の手順では、mysubnetgroup (コンソール)および AWS CLI というサブネットグループを作成する方法を示します。

サブネットグループの作成 (コンソール)

次の手順では、サブネットグループ (コンソール) を作成する方法を示します。

サブネットグループ (コンソール) を作成するには

1. AWS マネジメントコンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションリストで [Subnet Groups] を選択します。
3. Create Subnet Group を選択します。
4. Create Subnet Group ウィザードで、次の操作を行います。すべての設定が正しいことを確認したら、[Yes, Create] を選択します。
 - a. Name ボックスにサブネットグループの名前を入力します。
 - b. Description ボックスにサブネットグループの説明を入力します。
 - c. [VPC ID] ボックスで、作成した Amazon VPC を選択します。
 - d. [Availability Zone] および [Subnet ID] リストで、プライベートサブネットの Availability Zone または [Local Zone](#) と ID を選択し、[Add] を選択します。

Subnet group settings

A subnet group is a collection of subnets (typically private). Designate a subnet group for your clusters running in an Amazon Virtual Private Cloud (VPC) environment.

Name

The name is required, can have up to 255 characters, and must begin with a letter. It should not end with a hyphen or contain two consecutive hyphens. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Description - optional

VPC ID
 The identifier for the VPC environment where your cluster is to run.
 [Create VPC](#)

For Multi-AZ high availability mode, choose IDs for at least two subnets from two Availability Zones in the table below.

Selected subnets (6) [Manage](#)

Availability Zone ▲	Subnet ID ▼	Outpost ID ▼	CIDR block ▼
us-east-1a	subnet- <input type="text"/>		172.31.16.0/20
us-east-1b	subnet- <input type="text"/>		172.31.32.0/20
us-east-1c	subnet- <input type="text"/>		172.31.0.0/20
us-east-1d	subnet- <input type="text"/>		172.31.80.0/20

5. 表示された確認メッセージで、Close を選択します。

ElastiCache コンソールの [サブネットグループ] のリストに新しいサブネットグループが表示されます。ウィンドウの下部で、サブネットグループを選択して、ウィンドウの下部で詳細 (このグループに関連付けられているすべてのサブネットなど) を確認します。

サブネットグループを作成する (AWS CLI)

コマンドプロンプトで、`create-cache-subnet-group` コマンドを使用してサブネットグループを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-subnet-group \
  --cache-subnet-group-name mysubnetgroup \
  --cache-subnet-group-description "Testing" \
```

```
--subnet-ids subnet-53df9c3a
```

Windows の場合:

```
aws elasticache create-cache-subnet-group ^  
  --cache-subnet-group-name mysubnetgroup ^  
  --cache-subnet-group-description "Testing" ^  
  --subnet-ids subnet-53df9c3a
```

このコマンドでは、次のような出力が生成されます。

```
{  
  "CacheSubnetGroup": {  
    "VpcId": "vpc-37c3cd17",  
    "CacheSubnetGroupDescription": "Testing",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-53df9c3a",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2a"  
        }  
      }  
    ],  
    "CacheSubnetGroupName": "mysubnetgroup"  
  }  
}
```

詳細については、AWS CLI のトピック「[create-cache-subnet-group](#)」を参照してください。

ステップ 2: クラスターを作成する

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

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

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

Redis (クラスターモードが無効) クラスターの作成 (コンソール)

ElastiCache コンソールを使用して Redis (クラスターモードが無効) クラスターを作成するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で Amazon ElastiCache コンソールを開きます。
2. 右上のリストから、このクラスターを起動する AWS リージョンを選択します。
3. ナビゲーションペインで、[Get started] (開始) を選択します。
4. [VPC の作成] を選択し、「[Virtual Private Cloud \(VPC\) の作成](#)」のステップに従います。
5. ElastiCache ダッシュボードページで Redis キャッシュを選択し、Redis キャッシュの作成を選択します。
6. [クラスター設定] で、以下を実行します。
 - a. [Configure and create a new cluster] (新しいクラスターを設定および作成) を選択します。
 - b. [Cluster mode] (クラスターモード) で、[Disabled] (無効) を選択します。
 - c. [Cluster info] (クラスター情報) で、[Name] (名前) の値を入力します。
 - d. (オプション) [Description] (説明) の値を入力します。
7. [Location] (場所):

AWS Cloud

1. [AWS Cloud] (AWS クラウド) の場合、[Multi-AZ] (マルチ AZ) および [Auto-failover] (自動フェイルオーバー) のデフォルト設定を受け入れることをお勧めします。詳細については、「[マルチ AZ を使用した for Redis のダウンタイムの最小化 ElastiCache](#)」を参照してください。
2. [Cluster settings] (クラスター設定)
 - a. [Engine version] (エンジンバージョン) で、使用可能なバージョンを選択します。
 - b. [Port] (ポート) で、デフォルトポート 6379 を使用します。異なるポートを使用する理由がある場合は、そのポート番号を入力します。
 - c. [パラメータグループ] で、パラメータグループを選択するか、新しいパラメータグループを作成します。パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「[Redis 固有のパラメータ](#)」および「[パラメータグループを作成する](#)」を参照してください。

Note

パラメータグループを選択してエンジン設定値を設定すると、そのパラメータグループが Global Datastore 内のすべてのクラスターに適用されます。[パラメータグループ] ページの yes/no [グローバル] 属性は、パラメータグループがグローバルデータストアの一部であるかどうかを示します。

- d. [ノードタイプ] で、下向き矢印



を選択します。[ノードタイプの変更] ダイアログボックスで、必要なノードタイプの [インスタンスファミリー] の値を選択します。次に、このクラスターで使用するノードタイプを選択し、[保存] を選択します。

詳細については、「[ノードサイズの選択](#)」を参照してください。

r6gd ノードタイプを選択すると、データ階層化が自動的に有効になります。詳細については、「[データ階層化](#)」を参照してください。

- e. [Number of replicas] (レプリケーション数) で、必要なリードレプリカの数を選択します。マルチ AZ を有効にした場合、数値は 1~5 の間である必要があります。

3. [Connectivity] (接続) で

- a. [Network type] (ネットワークタイプ) で、このクラスターがサポートする IP バージョンを選択します。
- b. サブネットグループで、このクラスターに適用するサブネットを選択します。はそのサブネットグループ ElastiCache を使用して、ノードに関連付けるサブネットと IP アドレスを選択します。ElastiCache クラスターには、デュアルスタックモードで動作するように割り当てられた IPv4 アドレスと IPv6 アドレスの両方を持つデュアルスタックサブネットと、IPv6-onlyとして動作するように IPv6-onlyサブネットが必要です。

新しいサブネットグループを作成するときは、そのサブネットグループが属する VPC ID を入力します。

詳細については、以下を参照してください。

- [ネットワークタイプの選択](#).
- [VPC にサブネットを作成します](#)。


[ElastiCache での Local Zones の使用](#) である場合は、ローカルゾーンにあるサブネットを作成または選択する必要があります。

詳細については、「[サブネットおよびサブネットグループ](#)」を参照してください。

4. [Availability zone placements] (アベイラビリティゾーンの配置) には 2 つのオプションがあります。
 - 設定なし — ElastiCache アベイラビリティゾーンを選択します。
 - [アベイラビリティゾーンの指定] – 各クラスターに対するアベイラビリティゾーンを指定します。

アベイラビリティゾーンの指定を選択した場合、クラスターのシャードごとにリストからアベイラビリティゾーンを選択します。

詳細については、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

5. [Next] (次へ) を選択します。
 6. [Advanced Redis settings] (Redis の詳細設定) で
 - [Security] (セキュリティ):
 - i. データを暗号化するには、次のオプションがあります。
 - 保管時の暗号化 – ディスクに保存されているデータの暗号化を有効にします。詳細については、「[保管時の暗号化](#)」を参照してください。
-  **Note**

カスタマーマネージド AWS KMS キーを選択し、そのキーを選択することで、別の暗号化キーを指定できます。詳細については、「[AWS KMS のカスタマー管理の CMK の使用](#)」を参照してください。
- [転送中の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「[転送中の暗号化](#)」を参照してください。Redis エンジンバージョン 6.0

以降では、転送中の暗号化を有効にすると、次のアクセスコントロールオプションのいずれかを指定するよう求められます。

- アクセスコントロールなし — これがデフォルトの設定です。これは、クラスターへのユーザーアクセスに制限がないことを示します。
- [ユーザーグループのアクセスコントロールリスト] — クラスターにアクセスできるユーザーのセットが定義されているユーザーグループを選択します。詳細については、「[コンソールおよび CLI を使用したユーザーグループの管理](#)」を参照してください。
- [Redis AUTH デフォルトユーザー] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。
- [Redis AUTH] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。

Note

3.2.6 以降の Redis バージョン (バージョン 3.2.10 を除く) では、Redis AUTH のみがオプションとなります。

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

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

7. 自動バックアップを定期的にスケジュールする場合は、[自動バックアップの有効化] を選択し、自動バックアップを保持して自動的に削除するまでの日数を入力します。自動バックアップを定期的にスケジュールしない場合は、[自動バックアップを有効化] チェックボックスをオフにします。いずれの場合も、常に手動バックアップを作成するオプションがあります。

Redis のバックアップと復元の詳細については、「[スナップショットおよび復元](#)」を参照してください。

8. (オプション) メンテナンスウィンドウを指定します。[メンテナンスウィンドウ] は、ElastiCache がクラスターのシステムメンテナンスを毎週スケジュールする時間の長

さ (通常は 1 時間単位) です。ElastiCache がメンテナンスの日時を選択することを許可するか ([No preference])、自分で日時と期間を選択できます ([Specify maintenance window])。メンテナンスウィンドウを指定を選択した場合は、リストからメンテナンス期間の Start day、開始時間および期間を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「[メンテナンスの管理](#)」を参照してください。

9. (オプション) [ログ]:

- [ログの形式] の下で、[テキスト] または [JSON] を選択します。
 - 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
 - ログ送信先 で、新規作成を選択し、CloudWatch ログログロググループ名または Firehose ストリーム名を入力するか、既存選択を選択して、CloudWatch ログロググループ名または Firehose ストリーム名を選択します。
10. タグ では、クラスターやその他の ElastiCache リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。詳細については、「[ElastiCache リソースのタグ付け](#)」を参照してください。
11. [次へ] をクリックします。
12. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、[Create] (作成) を選択します。

On premises

1. [On premises] (オンプレミス) では、[Auto-failover] (自動フェイルオーバー) を有効のままにしておくことをお勧めします。詳細については、「[マルチ AZ を使用した for Redis のダウンタイムの最小化 ElastiCache](#)」を参照してください。
2. クラスターの作成を完了するには、「[Outposts の使用](#)」の手順に従います。

クラスターのステータスが [available] になり次第、Amazon EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、「[ステップ 3: クラスターへのアクセスの許可](#)」および「[ステップ 4: クラスターのノードに接続する](#)」を参照してください。

⚠ Important

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

Redis (クラスターモードが無効) クラスターの作成 (AWS CLI)

Example

次の CLI コードでは、レプリカのない Redis (クラスターモードが無効) キャッシュクラスターを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-cluster \  
--cache-cluster-id my-cluster \  
--cache-node-type cache.r4.large \  
--engine redis \  
--num-cache-nodes 1 \  
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Windows の場合:

```
aws elasticache create-cache-cluster ^  
--cache-cluster-id my-cluster ^  
--cache-node-type cache.r4.large ^  
--engine redis ^  
--num-cache-nodes 1 ^  
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

有効になっているクラスターモードを使用するには、以下のトピックを参照してください。

- コンソールを使用するには、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。
- を使用するには AWS CLI、「」を参照してください [Redis \(クラスターモードが有効\) クラスターの作成 \(AWS CLI\)](#)。

ステップ 3: クラスターへのアクセスの許可

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

すべての ElastiCache クラスターは Amazon EC2 インスタンスからアクセスするように設計されています。最も一般的なシナリオは、同じ Amazon Virtual Private Cloud (Amazon VPC) 内の Amazon EC2 インスタンスから ElastiCache クラスターにアクセスすることであり、それがこの演習でのケースとなります。

デフォルトでは、クラスターへのネットワークアクセスは、クラスターの作成に使用されたアカウントに制限されます。EC2 インスタンスからクラスターに接続するには、EC2 インスタンスにクラスターへのアクセスを許可する必要があります。必要なステップはクラスターを EC2-VPC で起動したか、EC2-Classical で起動したかによって異なります。

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

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

Redis クラスターとレプリケーショングループのデフォルトのポートは 6379 です。

Important

Amazon ElastiCache セキュリティグループは、Amazon Virtual Private Cloud (VPC) 環境で実行されていないクラスターのみにも適用されます。Amazon Virtual Private Cloud で実行している場合、[セキュリティグループ] はコンソールのナビゲーションペインでは使用できません。

ElastiCache ノードを Amazon VPC で実行している場合は、Amazon VPC セキュリティグループでクラスターへのアクセスを制御します。これは、ElastiCache セキュリティグループとは異なります。Amazon VPC で ElastiCache を使用する方法については、「[Amazon VPC と ElastiCache のセキュリティ](#)」を参照してください。

2. EC2 インスタンス (ウェブサーバーとアプリケーションサーバー) 用の VPC セキュリティグループを作成します。このセキュリティグループは、必要に応じて VPC のルーティングテーブル

ルを介してインターネットから EC2 インスタンスへのアクセスを許可できます。例えば、ポート 22 経由で EC2 インスタンスへの TCP アクセスを許可するルールをこのセキュリティグループに設定できます。

3. EC2 インスタンス用に作成したセキュリティグループからの接続を許可するクラスターのセキュリティグループで、カスタムルールを作成します。これは、セキュリティグループのメンバーにクラスターへのアクセスを許可します。

Note

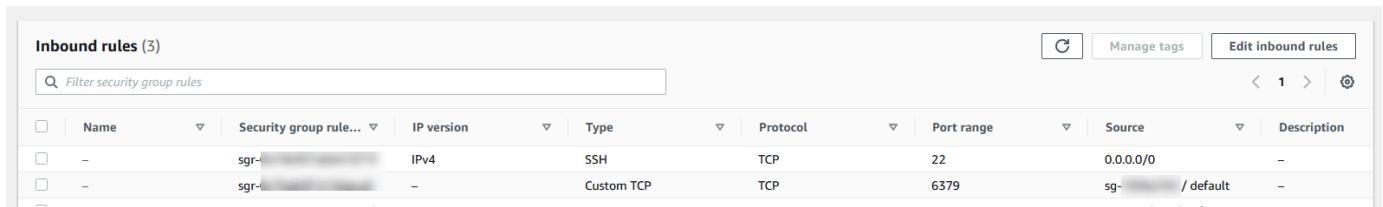
[[Local Zones](#)] の使用を計画している場合、それらが有効になっていることを確認します。そのローカルゾーンにサブネットグループを作成すると、VPC はそのローカルゾーンに拡張され、VPC はそのサブネットを他のアベイラビリティゾーンの子サブネットとして扱います。関連するすべてのゲートウェイとルートテーブルが自動的に調整されます。

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

1. AWS マネジメントコンソールにサインインして、Amazon VPC コンソール (<https://console.aws.amazon.com/vpc>) を開きます。
2. ナビゲーションペインで、[Security Groups] (セキュリティグループ) を選択します。
3. クラスターインスタンスに使用するセキュリティグループを選択または作成します。インバウンドルールで、インバウンドルールの編集を選択し、ルールの追加を選択します。このセキュリティグループは、他のセキュリティグループのメンバーへのアクセスを許可します。
4. Type で Custom TCP Rule を選択します。
 - a. Port Range ポートには、クラスター作成時に使用したポートを指定します。

Redis クラスターとレプリケーショングループのデフォルトのポートは 6379 です。

- b. ソースボックスに、セキュリティグループの ID の入力を開始します。リストから、Amazon EC2 インスタンスに使用するセキュリティグループを選択します。
5. 終了したら、保存を選択します。



<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-...	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-...	-	Custom TCP	TCP	6379	sg-... / default	-
<input type="checkbox"/>	-

アクセスを有効にしたので、次のセクションで説明するように、ノードに接続する準備が整いました。

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

- [Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)
- [AWS 外部からの ElastiCache リソースへのアクセス](#)

ステップ 4: クラスターのノードに接続する

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

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

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

ノードのエンドポイントを見つける

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

Redis (クラスターモードが無効) クラスターのエンドポイント (コンソール)

Redis (クラスターモードが無効) クラスターに 1 つのみのノードがある場合、ノードのエンドポイントは読み取りと書き込みの両方に使用されます。クラスターに複数のノードがある場合は、プライマリエンドポイント、リーダーエンドポイント、ノードエンドポイントの 3 種類のエンドポイントがあります。

プライマリエンドポイントは、常にクラスターのプライマリノードに解決される DNS 名です。プライマリエンドポイントは、リードレプリカのプライマリロールへの昇格など、クラスターに対する変更の影響を受けません。書き込みアクティビティの場合、アプリケーションをプライマリエンドポイントに接続することをお勧めします。

読み込みエンドポイントによって、ElastiCache for Redis クラスター内のすべてのリードレプリカ間でエンドポイントへの着信接続が均等に分割されます。アプリケーションがいつ接続を作成するか、アプリケーションが接続をどのように (再) 利用するかなどの追加要因によって、トラフィックの分散が決定されます。レプリカが追加または削除されても、読み込みエンドポイントはリアルタイムでクラスターの変更に対応します。ElastiCache for Redis クラスターの複数のリードレプリカを異なる AWS アベイラビリティーゾーン (AZ) に配置して、リーダーエンドポイントの高可用性を確保することができます。

Note

リーダーエンドポイントはロードバランサーではありません。これは、ラウンドロビン方式でレプリカノードの 1 つの IP アドレスに解決される DNS レコードです。

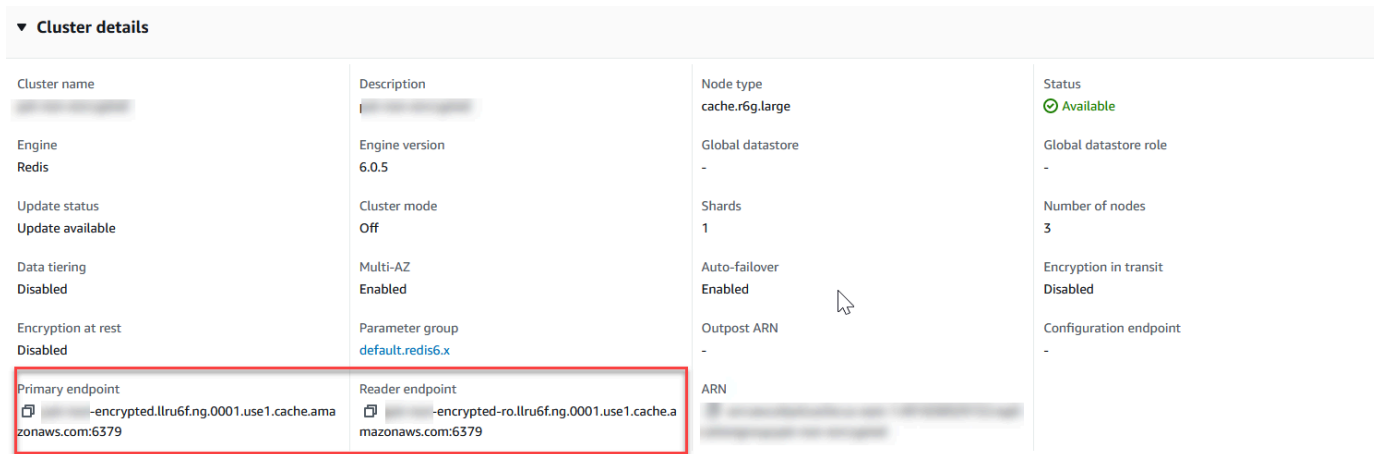
読み取りアクティビティの場合、アプリケーションはクラスター内のいずれのノードにも接続できます。プライマリエンドポイントとは異なり、ノードエンドポイントは特定のエンドポイントに解決されます。レプリカの追加または削除など、クラスターに変更を加えた場合は、アプリケーションでノードエンドポイントを更新する必要があります。

Redis (クラスターモードが無効) クラスターのエンドポイントを検索するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis キャッシュ] を選択します。

クラスター画面には 既存の Redis サーバーレスキャッシュ、Redis (クラスターモードが無効) クラスターと Redis (クラスターモードが有効) を含むリストが表示されます。[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#) のセクションで作成したものを選択します。

3. クラスターのプライマリエンドポイントやリーダーエンドポイントを検索するには、クラスターの名前 (ラジオボタンではない) を選択します。



▼ Cluster details			
Cluster name	Description	Node type cache.r6g.large	Status Available
Engine Redis	Engine version 6.0.5	Global datastore -	Global datastore role -
Update status Update available	Cluster mode Off	Shards 1	Number of nodes 3
Data tiering Disabled	Multi-AZ Enabled	Auto-failover Enabled	Encryption in transit Disabled
Encryption at rest Disabled	Parameter group default.redis6.x	Outpost ARN -	Configuration endpoint -
Primary endpoint [icon] [redacted]-encrypted.llru6f.ng.0001.use1.cache.amazonaws.com:6379	Reader endpoint [icon] [redacted]-encrypted-ro.llru6f.ng.0001.use1.cache.amazonaws.com:6379	ARN [redacted]	

Redis (クラスターモードが無効) クラスターのプライマリエンドポイント

クラスターに 1 つのみのノードがある場合、プライマリエンドポイントはないため、次のステップに進むことができます。

4. Redis (クラスターモードが無効) クラスターにレプリカノードがある場合は、クラスターの名前を選択してから [Nodes] (ノード) タブを選択して、クラスターのレプリカノードエンドポイントを検索できます。

ノードの画面では、クラスター内のプライマリとレプリカの各ノードがそのエンドポイントと共に表示されます。

- [エンドポイントの検索 \(AWS CLI\)](#)
- [エンドポイントの検索 \(ElastiCache API\)](#)

Redis クラスターまたはレプリケーショングループに接続する (Linux)

これで、必要なエンドポイントがわかったので、EC2 インスタンスにログインし、クラスターまたはレプリケーショングループに接続できます。次の例では、redis-cli ユーティリティを使用して、クラスターに接続します。redis-cli の最新バージョンでは、暗号化/認証が有効なクラスターを接続するための SSL/TLS もサポートしています。

次の例では、Amazon Linux および Amazon Linux 2 を実行している Amazon EC2 インスタンスを使用しています。他の Linux ディストリビューションでの redis-cli のインストールとコンパイルの詳細については、ご使用のオペレーティングシステムのドキュメントを参照してください。

Note

このプロセスでは、予期しない使用のみを対象として、redis-cli ユーティリティを使用した接続のテストをカバーしています。サポートされる Redis クライアントのリストについては、「[Redis ドキュメント](#)」を参照してください。AWS ElastiCache を用いた SDK の使用の例については、「[ElastiCache および AWS SDK の使用開始](#)」を参照してください。

クラスターモードが無効な非暗号化クラスターへの接続

1. 次のコマンドを実行してクラスターに接続し、`[primary-endpoint]` と `[#####]` をクラスターのエンドポイントとポート番号に置き換えます。(Redis のデフォルトポートは、6379 です。)

```
src/redis-cli -h primary-endpoint -p port number
```

次のような Redis のコマンドプロンプトが表示されます。

```
primary-endpoint:port number
```

2. Redis コマンドを実行できるようになりました。

```
set x Hello
OK
```

```
get x
"Hello"
```

クラスターモードが有効の非暗号化クラスターへの接続

1. 次のコマンドを実行してクラスターに接続し、`[configuration-endpoint]` と `[#####]` をクラスターのエンドポイントとポート番号に置き換えます。(Redis のデフォルトポートは、6379 です。)

```
src/redis-cli -h configuration-endpoint -c -p port number
```

Note

上記のコマンドで、`-c` オプションを指定すると、[-ASK および -MOVED リダイレクト](#)が続くクラスターモードが有効になります。

次のような Redis のコマンドプロンプトが表示されます。

```
configuration-endpoint:port number
```

2. Redis コマンドを実行できるようになりました。リダイレクトは、`-c` オプションを使用して有効にしたために発生します。リダイレクトが有効でない場合、このコマンドは MOVED エラーを返します。MOVED エラーの詳細については、「[Redis クラスター仕様](#)」を参照してください。

```
set x Hi
-> Redirected to slot [16287] located at 172.31.28.122:6379
OK
set y Hello
OK
get y
"Hello"
set z Bye
-> Redirected to slot [8157] located at 172.31.9.201:6379
OK
get z
"Bye"
get x
-> Redirected to slot [16287] located at 172.31.28.122:6379
```

```
"Hi"
```

暗号化/認証が有効なクラスターへの接続

デフォルトでは、redis-cli は Redis に接続するときに暗号化されていない TCP 接続を使用します。オプション BUILD_TLS=yes により、前の [redis-cli をダウンロードしてセットアップする](#) セクションで示されたように、redis-cli コンパイル時に SSL/TLS を有効にします。AUTH の有効化はオプションです。ただし、AUTH を有効にするために、転送時の暗号化を有効にする必要があります。ElastiCache の暗号化と認証の詳細については、「[ElastiCache 転送時の暗号化 \(TLS\)](#)」を参照してください。

Note

--tls redis-cli を用いたオプションを使用して、クラスターモードが有効および無効の両方の暗号化されたクラスターに接続できます。クラスターに AUTH トークンが設定されている場合は、オプション -a を使用して AUTH パスワードを指定します。

以下の例では、`[cluster-endpoint]` と `[#####]` をクラスターのエンドポイントとポート番号に置き換えてください。(Redis のデフォルトポートは、6379 です。)

クラスターモードが無効の暗号化されたクラスターに接続する

次の例では、暗号化および認証が有効のクラスターに接続します。

```
src/redis-cli -h cluster-endpoint --tls -a your-password -p port number
```

次の例では、暗号化のみが有効なクラスターに接続します。

```
src/redis-cli -h cluster-endpoint --tls -p port number
```

クラスターモードが有効の暗号化されたクラスターへの接続

次の例では、暗号化および認証が有効のクラスターに接続します。

```
src/redis-cli -c -h cluster-endpoint --tls -a your-password -p port number
```

次の例では、暗号化のみが有効なクラスターに接続します。

```
src/redis-cli -c -h cluster-endpoint --tls -p port number
```

クラスターに接続した後、暗号化されていないクラスターに対して、前の例に示されているように Redis コマンドを実行できます。

Redis-cli の代替

クラスターがクラスターモードが有効ではなく、短いテストのためにクラスターに接続する必要があるが、redis-cli コンパイルを行わない場合、telnet または openssl を使用できます。以下のコマンド例では、[*cluster-endpoint*] と [#####] をクラスターのエンドポイントとポート番号に置き換えてください。(Redis のデフォルトポートは、6379 です。)

次の例では、暗号化および/または認証が有効のクラスターモードが無効のクラスターに接続します。

```
openssl s_client -connect cluster-endpoint:port number
```

クラスターにパスワードが設定されている場合は、まずクラスターに接続します。接続後、次のコマンドを使用してクラスターを認証してから、Enter キーを押します。次の例では、[*your-password*] をクラスターのパスワードに置き換えます。

```
Auth your-password
```

次の例では、暗号化または認証が有効ではないクラスターモードが無効のクラスターに接続します。

```
telnet cluster-endpoint port number
```

Redis クラスターまたはレプリケーショングループに接続する (Windows)

Redis CLI を使用して EC2 Windows インスタンスから Redis クラスターに接続するには、redis-cli パッケージをダウンロードし、redis-cli.exe を使用して EC2 Windows インスタンスから Redis クラスターに接続する必要があります。

次の例では、redis-cli ユーティリティを使用して、暗号化が有効ではなく Redis を実行しているクラスターに接続します。Redis および使用可能な Redis コマンドの詳細については、Redis ウェブサイトで「[Redis のコマンド](#)」を参照してください。

redis-cli を使用して、暗号化が有効になっていない Redis クラスターに接続するには

1. 選択した接続ユーティリティを使用して、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方法については、「[Amazon EC2 入門ガイド](#)」を参照してください。
2. リンク <https://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zip> をコピーしてインターネットブラウザに貼り付け、Github の利用可能なリリース (<https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504>) から Redis クライアントの zip ファイルをダウンロードします。

zip ファイルを目的のフォルダ/パスに展開します。

コマンドプロンプトを開き、Redis ディレクトリに移動し、コマンド `c:\Redis>redis-cli -h Redis_Cluster_Endpoint -p 6379` を実行します。

例:

```
c:\Redis>redis-cli -h cmd.xxxxxxx.ng.0001.usw2.cache.amazonaws.com -p 6379
```

3. Redis コマンドを実行します。

これで、クラスターに接続され、以下のような Redis コマンドを実行できます。

```
set a "hello"           // Set key "a" with a string value and no expiration
OK
get a                   // Get value for key "a"
"hello"
get b                   // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
"Good-bye"
get b                   // Get value for key "b"
"Good-bye"

                        // wait >= 5 seconds

get b
(nil)                   // key has expired, nothing returned
quit                    // Exit from redis-cli
```


ステップ 5: クラスターを削除する

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

Warning

ElastiCache for Redis クラスターを削除しても、手動スナップショットは保持されます。クラスターを削除する前に最終スナップショットを作成することもできます。自動キャッシュスナップショットは保持されません。詳細については、「[スナップショットおよび復元](#)」を参照してください。

AWS Management Consoleの使用

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

クラスターを削除するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ElastiCache コンソールのダッシュボードで、[Redis] を選択します。

Redis を実行しているすべてのキャッシュのリストが表示されます。

3. 削除するクラスターを選択するには、クラスターのリストからそのクラスターの名前を選択します。この場合、[ステップ 2: クラスターを作成する](#) で作成したクラスターの名前です。

Important

ElastiCache コンソールから、一度に 1 つずつクラスターを削除できます。複数のクラスターを選択すると、削除オペレーションが無効になります。

4. アクションとして、Delete (削除) を選択します。
5. [クラスターの削除] 確認画面でクラスターの名前を入力し、[最終バックアップ] を選択します。次に、[削除] をクリックしてクラスターを削除するか、[キャンセル] をクリックしてクラスターを保持します。

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

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

AWS CLIの使用

次のコードでは、キャッシュクラスター `my-cluster` を削除します。この場合、`my-cluster` を、[ステップ 2: クラスターを作成する](#) で作成したクラスターの名前に置き換えます。

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

`delete-cache-cluster` CLI アクションは 1 つのキャッシュクラスターのみを削除します。複数のキャッシュクラスターを削除する場合は、削除するキャッシュクラスターごとに `delete-cache-cluster` を呼び出します。1 つのキャッシュクラスターの削除が終了するまで待たなくても次のクラスターを削除できます。

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-cluster \  
  --cache-cluster-id my-cluster \  
  --region us-east-2
```

Windows の場合:

```
aws elasticache delete-cache-cluster ^  
  --cache-cluster-id my-cluster ^  
  --region us-east-2
```

詳細については、「AWS CLI for ElastiCache トピック [delete-cache-cluster](#)」を参照してください。

ElastiCache のチュートリアルと動画

次のチュートリアルでは、Amazon ElastiCache ユーザーにとって興味深いタスクを紹介します。

- [ElastiCache の動画](#)
- [チュートリアル: Amazon VPC の Amazon ElastiCache にアクセスする Lambda 関数の設定](#)

ElastiCache の動画

Amazon ElastiCache の基本的な概念と高度な概念を理解できる動画を以下で確認できます。AWS トレーニングの詳細については、「[AWS トレーニングと認定](#)」を参照してください。

トピック

- [入門者向け動画](#)
- [上級者向けの動画](#)

入門者向け動画

次の動画では、Amazon ElastiCache について紹介しています。

トピック

- [AWS re:Invent 2020: Amazon ElastiCache の新機能](#)
- [AWS re:Invent 2019: Amazon ElastiCache の新機能](#)
- [AWS re:Invent 2017: Amazon ElastiCache の新機能](#)
- [DAT204 — AWS NoSQL サービスでのスケーラブルなアプリケーションの構築 \(re:Invent 2015\)](#)
- [DAT207 — Amazon ElastiCache によるアプリケーションパフォーマンスの向上 \(AWS re:Invent 2013\)](#)

[AWS re:Invent 2020: Amazon ElastiCache の新機能](#)

[AWS re:Invent 2020: Amazon ElastiCache の新機能](#)

[AWS re:Invent 2019: Amazon ElastiCache の新機能](#)

[AWS re:Invent 2019: Amazon ElastiCache の新機能](#)

[AWS re:Invent 2017: Amazon ElastiCache の新機能](#)

[AWS re:Invent 2017: Amazon ElastiCache の新機能](#)

[DAT204 — AWS NoSQL サービスでのスケーラブルなアプリケーションの構築 \(re:Invent 2015\)](#)

このセッションでは、NoSQL データベースの利点について説明し、AWS が提供する主な NoSQL サービスである Amazon DynamoDB と Amazon ElastiCache の概要を確認します。その後、当社の

主要なお客様である Expedia と Mapbox を取り上げ、ユースケースやアーキテクト上の課題、さらには AWS NoSQL サービスを利用した対処方法について、設計パターンやベストプラクティスの情報を交えながら確認します。このセッションを通じて、NoSQL とその強力な機能についての理解を深め、データベース関連の課題に自信を持って対処できるようになります。

[DAT204 — AWS NoSQL サービスでのスケーラブルなアプリケーションの構築 \(re:Invent 2015\)](#)

DAT207 — Amazon ElastiCache によるアプリケーションパフォーマンスの向上 (AWS re:Invent 2013)

この動画では、Amazon ElastiCache を使用してインメモリキャッシングシステムを簡単にデプロイして、アプリケーションパフォーマンスを高める方法について学習できます。Amazon ElastiCache を使用してアプリケーションのレイテンシーを改善し、データベースサーバーの負荷を下げる方法について説明します。また、アプリケーションが増加しても管理とスケーリングが簡単なキャッシュ Layer を構築する方法も示します。このセッション中、キャッシュを有効にすることによりメリットが得られるさまざまなシナリオとユースケースについて調べ、Amazon ElastiCache に用意されている機能について説明します。

[DAT207 - Amazon ElastiCache によるアプリケーションパフォーマンスの向上 \(re:Invent 2013\)](#)

上級者向けの動画

次の動画では、Amazon ElastiCache に関するより高度なトピックも取り上げています。

トピック

- [Amazon ElastiCache ベストプラクティスによる成功のための設計 \(re: Invent 2020\)](#)
- [Amazon ElastiCache \(re: Invent 2019\) でリアルタイムアプリを強化する](#)
- [ベストプラクティス: Amazon EC2 から ElastiCache への Redis クラスターの移行 \(re: Invent 2019\)](#)
- [Amazon ElastiCache & Amazon Aurora STP11 によるファンタジースポーツプラットフォームのスケーリング \(re: Invent 2018\)](#)
- [Amazon ElastiCache を使用したクラウドでの信頼性とスケーラビリティのある Redis \(re: Invent 2018\)](#)
- [ElastiCache の詳しい説明: インメモリデータストアのデザインパターン \(re: Invent 2018\)](#)
- [DAT305 — Amazon ElastiCache の詳しい説明 \(re: Invent 2017\)](#)
- [DAT306 — Amazon ElastiCache の詳しい説明 \(re: Invent 2016\)](#)
- [DAT317 — IFTTT が ElastiCache for Redis を使用してイベントを予測する方法 \(re: Invent 2016\)](#)

- [DAT407 — Amazon ElastiCache の詳しい説明 \(re: Invent 2015\)](#)
- [SDD402 — Amazon ElastiCache の詳細 \(re: Invent 2014 \)](#)
- [DAT307 - Amazon ElastiCache のアーキテクチャおよびデザインパターンの詳しい説明 \(re:Invent 2013\)。](#)

Amazon ElastiCache ベストプラクティスによる成功のための設計 (re: Invent 2020)

Redis 上に構築されたビジネスクリティカルなリアルタイムアプリケーションの爆発的な増加に伴い、可用性、スケーラビリティ、およびセキュリティが最も重要な考慮事項となっています。Amazon ElastiCache をオンラインスケールリング、マルチ AZ デプロイ全体にわたる高可用性、およびセキュリティ設定で正常にセットアップするためのベストプラクティスを学びます。

[Amazon ElastiCache ベストプラクティスによる成功のための設計 \(re: Invent 2020\)](#)

Amazon ElastiCache (re: Invent 2019) でリアルタイムアプリを強化する

クラウド導入の急速な成長と、新しいシナリオにより、アプリケーションには 1 秒間に数百万件のリクエストをサポートするために、マイクロ秒のレイテンシーと高いスループットが必要です。デベロッパーは、従来、リアルタイムアプリケーションのデータを管理するために、データ削減技術と組み合わせたディスクベースのデータベースのような、特殊なハードウェアと回避策に頼ってきました。これらのアプローチは高価で、スケーラブルではありません。フルマネージド型のインメモリ Amazon ElastiCache を使用して、究極のパフォーマンス、高いスケーラビリティ、可用性、およびセキュリティを実現し、リアルタイムアプリケーションのパフォーマンスを向上させる方法について説明します。

[Amazon ElastiCache \(re: Invent 2019:\) でリアルタイムアプリを強化する](#)

ベストプラクティス: Amazon EC2 から ElastiCache への Redis クラスターの移行 (re: Invent 2019)

Redis クラスターを自分で管理するのは難しい場合があります。ハードウェアのプロビジョニング、ソフトウェアのパッチ適用、データのバックアップ、およびワークロードのモニタリングを常に行う必要があります。Amazon ElastiCache 用に新しくリリースされたオンライン移行機能により、Amazon EC2 のセルフホスト型 Redis から、クラスターモードを無効にして、完全マネージド型 Amazon ElastiCache にデータを簡単に移動できるようになりました。このセッションでは、新しいオンライン移行ツールについて学び、デモを参照し、さらに重要なことは、Amazon ElastiCache へのスムーズな移行のための実践的なベストプラクティスを学習します。

[ベストプラクティス: Amazon EC2 から ElastiCache への Redis クラスターの移行 \(re: Invent 2019\)](#)

Amazon ElastiCache & Amazon Aurora STP11 によるファンタジースポーツプラットフォームのスケーリング (re: Invent 2018)

Dream11は、インド有数のスポーツテクノロジーのスタートアップ企業です。同社は、ファンタジークリケット、サッカー、バスケットボールなどの複数のスポーツをプレイする 4000 万人以上のユーザーの成長基盤を持っており、現在、50 ミリ秒の応答時間で毎分 300 万リクエストを生成する 100 万人の同時ユーザーにサービスを提供しています。このトークでは、Dream11 CTO の Amit Sharma 氏が、Amazon Aurora と Amazon ElastiCache を使用してフラッシュトラフィックを処理する方法について説明しています。フラッシュトラフィックは 30 秒の応答枠内で 3 倍になることがあります。Sharma 氏は、ロックなしでトランザクションをスケーリングすることについても語っています。また、フラッシュトラフィックを処理するためのステップを共有し、毎日アクティブユーザーに 500 万人にサービスを提供しています。完全なタイトル: AWS re:Invent 2018: Amazon ElastiCache & Amazon Aurora でファンタジースポーツプラットフォームをスケーリングする (STP11)

[Amazon ElastiCache & Amazon Aurora STP11 によるファンタジースポーツプラットフォームのスケーリング \(re: Invent 2018\)](#)

Amazon ElastiCache を使用したクラウドでの信頼性とスケーラビリティのある Redis (re: Invent 2018)

このセッションでは、Redis 互換サービスである Amazon ElastiCache for Redis の機能と機能強化について説明します。Redis 5、スケーラビリティとパフォーマンスの向上、セキュリティとコンプライアンスなど、主要な機能をカバーしています。また、今後の機能とお客様のケーススタディについても説明します。

[Amazon ElastiCache を使用したクラウドでの信頼性とスケーラビリティのある Redis \(re: Invent 2018\)](#)

ElastiCache の詳しい説明: インメモリデータストアのデザインパターン (re: Invent 2018)

このセッションでは、Amazon ElastiCache の設計とアーキテクチャについて学ぶために、その裏側をのぞいてみます。Redis や Memcached の一般的な設計パターンを示すと共に、お客様がこれらをメモリ内のデータ処理でどのように使用し、レイテンシーを減らしてアプリケーションスループットを向上させたかを説明します。ElastiCache のベストプラクティス、設計パターン、およびアンチパターンについて概説します。

[ElastiCache の詳しい説明: インメモリデータストアのデザインパターン \(re: Invent 2018\)](#)

[DAT305 — Amazon ElastiCache の詳しい説明 \(re: Invent 2017\)](#)

Amazon ElastiCache の設計やアーキテクチャを理解するために、その裏側を見てみましょう。Memcached や Redis の一般的な設計パターンを示すと共に、お客様がこれらをメモリ内のオペレーションでどのように使用し、レイテンシーを減らしてアプリケーションスループットを向上させたかを説明します。このビデオでは、ElastiCache のベストプラクティス、設計パターン、およびアンチパターンについて概説します。

ビデオでは次の機能を紹介しています。

- ElastiCache for Redis のオンラインリシャーディング
- ElastiCache のセキュリティと暗号化
- ElastiCache for Redis バージョン 3.2.10

[DAT305 — Amazon ElastiCache の詳しい説明 \(re: Invent 2017\)](#)

[DAT306 — Amazon ElastiCache の詳しい説明 \(re: Invent 2016\)](#)

Amazon ElastiCache の設計やアーキテクチャを理解するために、その裏側を見てみましょう。Memcached や Redis の一般的な設計パターンを示すと共に、お客様がこれらをメモリ内のオペレーションでどのように使用し、レイテンシーを減らしてアプリケーションスループットを向上させたかを説明します。このセッションでは、ElastiCache のベストプラクティス、設計パターン、およびアンチパターンについて概説します。

[DAT306 — Amazon ElastiCache の詳しい説明 \(re: Invent 2016\)](#)

[DAT317 — IFTTT が ElastiCache for Redis を使用してイベントを予測する方法 \(re: Invent 2016\)](#)

IFTTT は、簡単なタスクの自動化から、家での操作と制御の方法の転換まで、ユーザーが好みのサービスでよりさまざまなことができるように強化する無料サービスです。IFTTT は ElastiCache for Redis を使用して、トランザクション実行履歴の保存、スケジュールの予測を行っています。また、Amazon S3 でログドキュメントのインデックス付けも行っています。このセッションを表示して、Lua のスクリプトのパワーと Redis のデータ型によって他ではできないことを実現できた方法を学んでください。

[DAT317 — IFTTT が ElastiCache for Redis を使用してイベントを予測する方法 \(re: Invent 2016\)](#)

[DAT407 — Amazon ElastiCache の詳しい説明 \(re: Invent 2015\)](#)

Amazon ElastiCache の設計やアーキテクチャを理解するために、その裏側をのぞいてみましょう。まず、Memcached や Redis の一般的な設計パターンを示すと共に、お客様がこれらをメモリ内の操

作でどのように使用し、アプリケーションのレイテンシーやスループットをどのように向上させてきたかを説明します。このセッションでは、Amazon ElastiCache に関連するベストプラクティス、設計パターン、アンチパターンについて概説します。

[DAT407 — Amazon ElastiCache の詳しい説明 \(re: Invent 2015\)](#)

SDD402 — Amazon ElastiCache の詳細 (re: Invent 2014)

この動画では、一般的なキャッシュのユースケース、Memcached エンジンと Redis エンジン、ニーズに合うエンジンを判断するのに役立つパターン、整合性のあるハッシュ、高速でスケーラブルなアプリケーションを構築する他の手段について調べます。Adobe が Amazon ElastiCache を使用してカスタマーエクスペリエンスを高め、ビジネスを拡大している方法について、Adobe のプリンシパルサイエンティストである Frank Wiebe 氏が詳しく説明します。

[DAT402 — Amazon ElastiCache の詳しい説明 \(re: Invent 2014\)](#)

DAT307 - Amazon ElastiCache のアーキテクチャおよびデザインパターンの詳しい説明 (re:Invent 2013)。

この動画では、キャッシュ、キャッシュ戦略、拡張、モニタリングについて検討します。また、Memcached エンジンと Redis エンジンの比較も行います。このセッション中、Amazon ElastiCache に関連するベストプラクティスとデザインパターンも確認します。

[DAT307 - Amazon ElastiCache Architecture のアーキテクチャおよびデザインパターンの詳しい説明 \(AWS re:Invent 2013\)](#)

次のステップ

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

- [AWS の開始方法](#)
- [Amazon Web Services のツール](#)
- [AWS Command Line Interface](#)
- [Amazon ElastiCache API リファレンス](#)

開始方法の演習を完了したら、以下のセクションで ElastiCache の管理に関する詳細を確認できます。

- [ノードサイズの選択](#)

キャッシュは、キャッシュしたいすべてのデータに対応できるだけの十分な大きさにします。同時に、必要以上にキャッシュを大きくしたくはないものです。このトピックを使って、最良のノードサイズを選択します。

- [ElastiCache ベストプラクティスとキャッシュ戦略](#)

クラスターの効率に影響を及ぼす可能性がある問題を特定し、対処します。

ノードの管理

ノードとは、Amazon ElastiCache のデプロイにおける最小の構成要素です。これは、安全なネットワークに接続された RAM の固定サイズの断片です。各ノードは、クラスターまたはレプリケーショングループの作成時または最終変更時に選択されたエンジンを実行します。各ノードはそれぞれ Domain Name Service (DNS) 名とポートを持っています。複数のタイプの ElastiCache ノードがサポートされており、関連付けられたメモリ量と計算能力がそれぞれ異なります。

一般に、シャーディングがサポートされているため、Redis (クラスターモードが有効) のデプロイでは、クラスター内に多数の小さいノードが含まれます。対照的に、Redis (クラスターモードが無効) のデプロイでは、クラスター内に少数の大きいノードが含まれます。使用するノードサイズの詳細な説明については、「[ノードサイズを選択](#)」を参照してください。

トピック

- [ElastiCache ノードステータスの表示](#)
- [Redis のノードとシャード](#)
- [ノードに接続する](#)
- [サポートされているノードの種類](#)
- [ノードの再起動 \(クラスターモードが無効な場合のみ\)](#)
- [ノードの置換](#)
- [ElastiCache のリザーブドノード](#)
- [以前の世代のノードの移行](#)

ノードに関係するいくつかの重要なオペレーションは以下のとおりです。

- [クラスターへのノードの追加](#)
- [クラスターからのノードの削除](#)

- [Redis ElastiCache のスケーリング](#)
- [接続エンドポイントの検索](#)

ElastiCache ノードステータスの表示

[ElastiCache コンソールを使用すると](#)、ElastiCache ノードのステータスにすばやくアクセスできます。ElastiCache ノードのステータスはノードの状態を示します。以下の手順を使用して、Amazon ElastiCache コンソール、AWS CLI コマンド、または API ElastiCache オペレーションでノードステータスを表示できます。

ElastiCache ノードに設定できるステータス値は次の表のとおりです。この表には、ElastiCache そのノードの料金が請求されるかどうかも示されています。

タイプ	請求対象	説明
available	請求対象	ElastiCache ノードは正常で、使用可能です。
creating	非請求対象	ElastiCache ノードは作成中です。ノードの作成中はノードにアクセスできません。
deleting	非請求対象	ElastiCache ノードは削除中です。
modifying	請求対象	ElastiCache お客様からノードの変更依頼があったため、ノードを変更中です。
updating	請求対象	更新中状態は、Amazon ElastiCache ノードについて次の 1 つ以上が当てはまることを示します。 <ul style="list-style-type: none">• ElastiCache サービス更新の一環としてノードにパッチが適用されています。サービスアップデートの

タイプ	請求対象	説明
		<p>詳細については、Amazon ElastiCache マネージドメンテナンスとサービスアップデートのヘルプページを参照してください。</p> <ul style="list-style-type: none"> • VPC ElastiCache セキュリティグループがクラスター用に更新されています。 • ElastiCache クラスターはスケールアップまたはスケールダウン中です。 • クラスターのログ配信設定が変更されています。 <p>ElastiCache</p> <ul style="list-style-type: none"> • ElastiCache ノードの削除操作は保留中です。 • ElastiCache for Redis のパスワードは、を使用して更新/ローテーションされています。AWS Secrets Manager
rebooting cache cluster nodes	請求対象	顧客からのリクエスト、ElastiCache またはノードの再起動を必要とする Amazon ElastiCache プロセスにより、ノードを再起動中です。

タイプ	請求対象	説明
incompatible_parameters	非請求対象	<p>ノードのパラメータグループで指定されたパラメータがノードと互換性がないため、Amazon ElastiCache はノードを起動できません。パラメータの変更を元に戻すか、パラメータにノードとの互換性を持たせて、ノードへのアクセスを回復してください。互換性のないパラメータの詳細については、ElastiCache ノードのイベントリストを確認してください。</p>
incompatible_network	非請求対象	<p>互換性のないネットワーク状態は、Amazon ノードに次の1つ以上の条件に当てはまることを示します。ElastiCache</p> <ul style="list-style-type: none">• ElastiCache ノードが起動されたサブネットには使用可能な IP アドレスがありません。• ElastiCache サブネットグループに記載されているサブネットは、Amazon Virtual Private Cloud (Amazon VPC) にはもう存在しません。

タイプ	請求対象	説明
restore_failed	非請求対象	<p>復元に失敗した状態は、Amazon ノードが以下のいずれかに該当することを示します。ElastiCache</p> <ul style="list-style-type: none">• インスタンスの容量不足が原因で、ノードの交換が繰り返し失敗した。これは通常、前世代のノードを実行しているときに発生します。end-of-lifeただし、AWS 指定されたアベイラビリティゾーンでリクエストを満たすのに十分なオンデマンドキャパシティがない場合に、現世代のノードを交換して発生することもあります。これらのノードの修正または削除の詳細については、「」を参照してください以前の世代のノードの移行。• 指定した RDB スナップショットを復元できなかった。• AWS ElastiCache クラスターのアカウントは停止されています。• ノードに障害が発生し、復旧できませんでした。
snapshotting	請求対象	ElastiCache ElastiCache for Redis ノードのスナップショットを作成しています。

ElastiCache コンソールでノードステータスを表示する

ElastiCache コンソールでノードのステータスを表示するには:

1. AWS Management Console にサインインし、<https://console.aws.amazon.com/elasticache/> にある Amazon ElastiCache コンソールを開きます。
2. ナビゲーションペインで [Redis クラスター] または [Memcached クラスター] を選択します。[キャッシュ] ページに、ノードのリストが表示されます。ElastiCache ノードごとに、ステータス値が表示されます。
3. その後、キャッシュの「Service Updates」タブに移動して、そのキャッシュに適用可能なサービスアップデートのリストを表示できます。

ElastiCache でノードステータスを表示する AWS CLI

ElastiCache を使用してノードとそのステータス情報を表示するには AWS CLI、`describe-cache-cluster` コマンドを使用します。たとえば、AWS CLI ElastiCache 次のコマンドは各ノードを表示します。

```
aws elasticache describe-cache-clusters
```

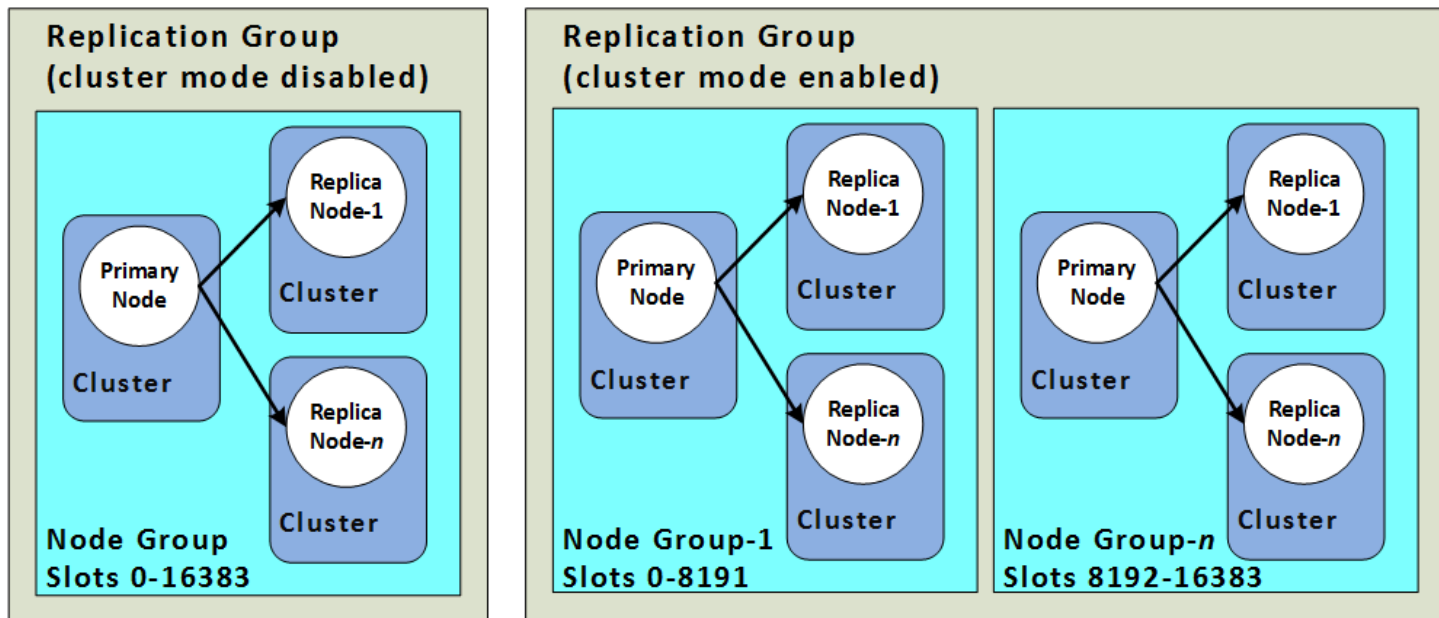
API ElastiCache によるノードステータスの表示

Amazon ElastiCache API ElastiCache を使用してノードのステータスを表示するには、`DescribeCacheClusterOperationShowCacheNodeInfo` フラグを付けて呼び出し、個々のキャッシュノードに関する情報を取得します。

Redis のノードとシャード

シャード (API と CLI ではノードグループ) はノードの階層的配列であり、各ノードはクラスターに含まれます。シャードはレプリケーションをサポートします。シャード内では、1つのノードが読み取り/書き込みのプライマリノードとなります。他のすべてのノードは、プライマリノードの読み取り専用のレプリカとなります。Redis バージョン 3.2 以降は、クラスター内の複数のシャード (API および CLI ではレプリケーショングループ) をサポートします。このサポートにより、Redis (クラスターモードが有効) クラスター内でデータを分割できます。

次の図は、Redis (クラスターモードが無効) クラスターと Redis (クラスターモードが有効) クラスターの違いを示しています。



Redis (クラスターモードが有効) クラスターは、シャードを介したレプリケーションをサポートします。API オペレーションの [DescribeReplicationGroups](#) (CLI: [describe-replication-groups](#)) を使うと、ノードグループとそれを構成するメンバーノード、ノードグループ内での各ノードの役割、およびその他の情報などを確認できます。

Redis クラスターを作成するときは、クラスタリングを有効にしてクラスターを作成するかどうかを指定します。Redis (クラスターモードが無効) に複数のシャードがあることは決してありません。シャードはリードレプリカノードを (合計で最大 5 つまで) 追加するか、削除することによって、水平方向にスケールすることができます。詳細については、「[レプリケーショングループを使用する高可用性](#)」、「[Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの追加](#)」または「[Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの削除](#)」を参照してください。Redis (クラスターモードが無効) クラスターは、ノードタイプを変更することで垂直方向にもスケールできます。詳細については、「[レプリカノードを含む Redis \(クラスターモードが無効\) クラスターのスケール](#)」を参照してください。

Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

Redis (クラスターモードが有効) クラスターは作成された後、変更 (スケールインまたはスケールアウト) できます。詳細については、[Redis ElastiCache のスケーリング](#) および [ノードの置換](#) を参照してください。

新しいクラスターを作成するときに、古いクラスターからのデータをシードして、空から開始しないようにすることができます。このアプローチは、クラスターグループに古いクラスターと同じ数のシャードがある場合にのみ機能します。これは、ノードタイプまたはエンジンバージョンの変更が必要な場合に便利です。詳細については、[手動バックアップの取得](#) および [バックアップから新しいキャッシュへの復元](#) を参照してください。

ノードに接続する

Redis クラスターのノードに接続を試みる前に、ノードのエンドポイントが必要です。エンドポイントを見つけるには、以下を参照してください。

- [Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) クラスターのエンドポイントを検索する \(コンソール\)](#)
- [エンドポイントの検索 \(AWS CLI\)](#)
- [エンドポイントの検索 \(ElastiCache API\)](#)

次の例では、redis-cli ユーティリティを使用して Redis を実行しているクラスターに接続します。

Note

Redis と使用可能な Redis のコマンドの詳細については、<http://redis.io/commands> ウェブページを参照してください。

redis-cli を使用して Redis クラスターに接続するには

1. 選択した接続ユーティリティを使用して、Amazon EC2 インスタンスに接続します。

Note

Amazon EC2 インスタンスに接続する方法については、「[Amazon EC2 入門ガイド](#)」を参照してください。

2. redis-cli をビルドするには、GNU Compiler Collection (gcc) をダウンロードしてインストールします。EC2 インスタンスのコマンドプロンプトで、以下のコマンドを入力します。確認のプロンプトが表示されたら、「y」と入力します。

```
sudo yum install gcc
```

以下のような出力が表示されます。

```
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
```

```
Resolving Dependencies
--> Running transaction check


...(output omitted)...

Total download size: 27 M
Installed size: 53 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): binutils-2.22.52.0.1-10.36.amzn1.x86_64.rpm      | 5.2 MB    00:00
(2/11): cpp46-4.6.3-2.67.amzn1.x86_64.rpm              | 4.8 MB    00:00
(3/11): gcc-4.6.3-3.10.amzn1.noarch.rpm                | 2.8 kB    00:00

...(output omitted)...

Complete!
```

3. redis-cli ユーティリティをダウンロードし、コンパイルします。このユーティリティは Redis ソフトウェアディストリビューションに含まれています。EC2 インスタンスのコマンドプロンプトで、以下のコマンドを入力します。

 Note

Ubuntu システムでは、make を実行する前に、make distclean を実行します。

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean      # ubuntu systems only
make
```

4. EC2 インスタンスのコマンドプロンプトで、次のコマンドを入力します。

```
src/redis-cli -c -h mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com -p 6379
```

以下のような Redis コマンドのプロンプトが表示されます。

```
redis mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 6379>
```

5. Redis コマンドを実行して接続をテストします。

これで、クラスターに接続され、Redis のコマンドを実行できます。以下に一部のコマンドとその Redis レスポンスの例を示します。

```
set a "hello"           // Set key "a" with a string value and no expiration
OK
get a                   // Get value for key "a"
"hello"
get b                   // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
get b
"Good-bye"

                        // wait 5 seconds

get b
(nil)                   // key has expired, nothing returned
quit                    // Exit from redis-cli
```

Secure Sockets Layer (SSL) 暗号化 (転送中の暗号化) が有効なノードまたはクラスターへの接続については、「[ElastiCache 転送時の暗号化 \(TLS\)](#)」を参照してください。

サポートされているノードの種類

ElastiCache では以下のノードタイプがサポートされています。一般に、現行世代のタイプは、以前の世代の同等タイプと比較した場合、メモリが多く処理能力が高くなっています。

各ノードタイプのパフォーマンスの詳細の詳細については、「[Amazon EC2 インスタンスタイプ](#)」を参照してください。

使用するノードサイズの詳細については、「[ノードサイズを選択](#)」を参照してください。

現行世代

旧世代の詳細については、「[旧世代のノード](#)」を参照してください。

Note

バースト可能なネットワークパフォーマンスのあるインスタンスタイプでは、ネットワーク I/O クレジットメカニズムを使用して、ベストエフォートベースでベースライン帯域幅を超えてバーストします。

[全般]

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	パフォーマンス帯域幅 (Gbps)
cache.m7g.large	6.2	N	N	N	0.937	12.5
cache.m7g.xlarge	6.2	Y	Y	Y	1.876	12.5
cache.m7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.m7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.m7g.8xlarge	6.2	Y	Y	Y	15	該当なし
cache.m7g.12xlarge	6.2	Y	Y	Y	22.5	該当なし

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m7g.16xlarge	6.2	Y	Y	Y	30	該当なし
cache.m6g.large	5.0.6	N	N	N	0.75	10.0
cache.m6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0
cache.m6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0
cache.m6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.m6g.8xlarge	5.0.6	Y	Y	Y	12	該当なし

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m6g.12xlarge	5.0.6	Y	Y	Y	20	該当なし
cache.m6g.16xlarge	5.0.6	Y	Y	Y	25	該当なし
cache.m5.large	3.2.4	N	N	N	0.75	10.0
cache.m5.xlarge	3.2.4	Y	N	N	1.25	10.0
cache.m5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.m5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m5.12xlarge	3.2.4	Y	Y	Y	12	該当なし
cache.m5.24xlarge	3.2.4	Y	Y	Y	25	該当なし
cache.m4.large	3.2.4	N	N	N	0.45	1.2
cache.m4.xlarge	3.2.4	Y	N	N	0.75	2.8
cache.m4.2xlarge	3.2.4	Y	Y	Y	1.0	10.0
cache.m4.4xlarge	3.2.4	Y	Y	Y	2.0	10.0

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m4.10xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.t4g.micro	3.2.4	N	N	N	0.064	5.0
cache.t4g.small	5.0.6	N	N	N	0.128	5.0
cache.t4g.medium	5.0.6	N	N	N	0.256	5.0
cache.t3.micro	3.2.4	N	N	N	0.064	5.0
cache.t3.small	3.2.4	N	N	N	0.128	5.0
cache.t3.medium	3.2.4	N	N	N	0.256	5.0
cache.t2.micro	3.2.4	N	N	N	0.064	1.024
cache.t2.small	3.2.4	N	N	N	0.128	1.024
cache.t2.medium	3.2.4	N	N	N	0.256	1.024

メモリ最適化

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r7g.large	6.2	N	N	N	0.937	12.5
cache.r7g.xlarge	6.2	Y	Y	Y	1.876	12.5
cache.r7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.r7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.r7g.8xlarge	6.2	Y	Y	Y	15	該当なし
cache.r7g.12xlarge	6.2	Y	Y	Y	22.5	該当なし

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r7g.16xlarge	6.2	Y	Y	Y	30	該当なし
cache.r6g.large	5.0.6	N	N	N	0.75	10.0
cache.r6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0
cache.r6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0
cache.r6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.r6g.8xlarge	5.0.6	Y	Y	Y	12	該当なし

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r6g. .12xlarge	5.0.6	Y	Y	Y	20	該当なし
cache.r6g. .16xlarge	5.0.6	Y	Y	Y	25	該当なし
cache.r5.large	3.2.4	N	N	N	0.75	10.0
cache.r5.xlarge	3.2.4	Y	N	N	1.25	10.0
cache.r5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.r5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r5.12xlarge	3.2.4	Y	Y	Y	12	該当なし
cache.r5.24xlarge	3.2.4	Y	Y	Y	25	該当なし
cache.r4.large	3.2.4	N	N	N	0.75	10.0
cache.r4.xlarge	3.2.4	Y	N	N	1.25	10.0
cache.r4.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.r4.4xlarge	3.2.4	Y	Y	Y	5.0	10.0

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r4.8xlarge	3.2.4	Y	Y	Y	12	該当なし
cache.r4.16xlarge	3.2.4	Y	Y	Y	25	該当なし

データ階層化で最適化されたメモリ

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r6gd.xlarge	6.2.0	Y	N	N	1.25	10
cache.r6gd.2xlarge	6.2.0	Y	Y	Y	2.5	10
cache.r6gd.4xlarge	6.2.0	Y	Y	Y	5.0	10
cache.r6gd.8xlarge	6.2.0	Y	Y	Y	12	該当なし
cache.r6gd.12xlarge	6.2.0	Y	Y	Y	20	該当なし
cache.r6gd.16xlarge	6.2.0	Y	Y	Y	25	該当

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	ベースト帯域幅 (Gbps)
						なし

ネットワーク最適化

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	パフォーマンス帯域幅 (Gbps)
cache.c7gn.large	6.2	N	N	N	6.25	30
cache.c7gn.xlarge	6.2	Y	Y	Y	12.5	40
cache.c7gn.2xlarge	6.2	Y	Y	Y	25	50
cache.c7gn.4xlarge	6.2	Y	Y	Y	50	該当なし
cache.c7gn.8xlarge	6.2	Y	Y	Y	100	該当なし
cache.c7gn.12xlarge	6.2	Y	Y	Y	150	該当

インスタンスタイプ	サポートされている Redis の最小バージョン	拡張 I/O (Redis 5.0.6 以降)	TLS オフロード (Redis 6.2.5 以降)	拡張 I/O マルチプレクシング (Redis 7.0.4 以降)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
						なし
cache.c7g n.16xlarge	6.2	Y	Y	Y	200	該当なし

AWS リージョン別のサポート対象ノードタイプ

サポートされるノードタイプは、AWS リージョンによって異なる場合があります。詳細については、「[Amazon ElastiCache 料金表](#)」を参照してください。

バースト可能パフォーマンスインスタンス

Amazon ElastiCache では、バースト可能な汎用 T4g、T3-Standard および T2-Standard キャッシュノードを起動できます。これらのノードは、CPU パフォーマンスのベースラインレベルを提供し、発生したクレジットが使い果たされるまでいつでも CPU 使用率をバーストすることができます。CPU クレジットは、フル CPU パフォーマンスを 1 分間実現します。

Amazon ElastiCache の T4g、T3 および T2 ノードは、スタンダードとして設定され、CPU の平均使用率がインスタンスのベースラインパフォーマンスを一貫して下回るワークロードに適しています。ベースラインより上にバーストする場合、ノードは CPU クレジット残高に蓄積されたクレジットを消費します。蓄積されたクレジットで実行中のノードが少ない場合、パフォーマンスは次第にベースラインのパフォーマンスレベルに下がります。この段階的な低下により、蓄積された CPU クレジットバランスがなくなったときに、ノードでパフォーマンスの急激な低下が発生することがありません。詳細については、Amazon EC2 ユーザーガイドの「[バースト可能パフォーマンスインスタンスの CPU クレジットおよびベースラインパフォーマンス](#)」を参照してください。

次の表に、バースト可能なパフォーマンスのノードタイプと、1 時間あたりに受け取る CPU クレジットのレートを示します。また、ノードで蓄積可能な、受け取った CPU クレジットの最大数と、ノードあたりの vCPU の数も示します。さらに、完全なコアパフォーマンス (1 つの vCPU を使用) の割合として、ベースラインパフォーマンスレベルも示します。

1 時間あたりに受け取る CPU クレジット	蓄積可能な最大獲得クレジット*	vCPUs	vCPU 別のベースラインパフォーマンス	メモリ (GiB)	ネットワークパフォーマンス
12	288	2	10%	0.5	最大 5 ギガビット
24	576	2	20%	1.37	最大 5 ギガビット
24	576	2	20%	3.09	最大 5 ギガビット
12	288	2	10%	0.5	最大 5 ギガビット
24	576	2	20%	1.37	最大 5 ギガビット
24	576	2	20%	3.09	最大 5 ギガビット

1 時間あたりに受け取る CPU クレジット	蓄積可能な最大獲得クレジット*	vCPUs	vCPU 別のベースラインパフォーマンス	メモリ (GiB)	ネットワークパフォーマンス
6	144	1	10%	0.5	低から中程度
12	288	1	20%	1.55	低から中程度
24	576	2	20%	3.22	低から中程度

* 蓄積できるクレジットの数は、24 時間で獲得できるクレジットの数と同じです。

** このテーブルのベースラインのパフォーマンスは vCPU 別です。一部のノードサイズでは、vCPU の数は 1 より大きくなります。それらのノードサイズでは、ノードのベースライン CPU 使用率を計算するには、vCPU のパーセントを vCPU の数で乗算します。

次の CPU クレジットメトリクスは、T3 および T4g バーストパフォーマンスインスタンスで利用可能です。

Note

これらのメトリクスは、T2 バーストパフォーマンスインスタンスでは利用できません。

- CPUCreditUsage
- CPUCreditBalance

これらのメトリクスの詳細については、「[CPU クレジットメトリクス](#)」を参照してください。

また、以下の詳細についても注意してください。

- 現在のすべての世代のノードタイプは、デフォルトでは Amazon VPC に基づいて Virtual Private Cloud (VPC) で作成されます。

- Redis AOF (Append-Only File) は、T2 インスタンスではサポートされません。Redis 設定変数 `appendonly` および `appendfsync` Redis バージョン 2.8.22 以降ではサポートされていません。

関連情報

- [Amazon ElastiCache 製品の特徴と詳細](#)
- [Redis 固有のパラメータ](#)
- [転送時の暗号化 \(TLS\)](#)

ノードの再起動 (クラスターモードが無効な場合のみ)

変更内容によっては、変更を適用するためにクラスターノードを再起動する必要があります。例えば、一部のパラメータで、パラメータグループのパラメータ値の変更は、再起動後にのみ適用されません。

Redis (クラスターモードが無効) クラスターの場合、これらのパラメータは次のとおりです。

- アクティブハッシュ化
- databases

ElastiCache コンソールを使用してのみ、ノードを再起動できます。一度に再起動できるノードは 1 つだけです。複数のノードを再起動するには、ノードごとにプロセスを繰り返す必要があります。

Redis (クラスターモードが有効) パラメータの変更

Redis (クラスターモードが有効) クラスターで次のパラメータを変更する場合は、以降のステップに従います。

- アクティブハッシュ化
- databases

1. クラスターの手動バックアップを作成します。「[手動バックアップの取得](#)」を参照してください。
2. Redis (クラスターモードが有効) クラスターを削除します。「[クラスターの削除](#)」を参照してください。

3. 変更されたパラメータグループとバックアップを使用してクラスターを復元し、新しいクラスターをシードします。「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

他のパラメータを変更する場合、これは必要ありません。

AWS Management Console を使用する場合

ElastiCache コンソールを使用して、ノードを再起動できます。

ノードを再起動するには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 右上隅にあるリストから、AWS 適用されるリージョンを選択します。
3. 左のナビゲーションペインで、[Redis] を選択します。

Redis を実行しているクラスターのリストが表示されます。

4. [クラスター名] の下で、クラスターを選択します。
5. [ノード名] の下で、再起動するノードの横にあるラジオボタンを選択します。
6. [アクション]、[ノードの再起動] の順に選択します。

複数のノードを再起動するには、再起動するノードごとにステップ 2~5 を繰り返します。1 つのノードの再起動が終了するまで待たなくても、別のノードを再起動できます。

ノードの置換

Amazon ElastiCache for Redis は、インスタンスにシームレスに適用されるパッチとアップグレードを使用して、頻繁にフリートのアップグレードを行います。ただし、場合によっては基盤となるホスト OS の必須更新を適用するために ElastiCache for Redis ノードを再起動する必要があります。セキュリティ、信頼性、運用パフォーマンスを強化するアップグレードを適用するため、そのような置換が必要となります。

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

Note

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

次のリストで、ElastiCache が Redis ノードの 1 つの置き換えをスケジュールしている場合にとることができるアクションを識別します。状況に応じて必要となる情報をすばやく見つけるには、次のメニューから選択します。

- [Do nothing](#) — Amazon ElastiCache がスケジュールどおりにノードを置き換えるようにします。
- [Change your maintenance window](#) – メンテナンス時間をより適切な時刻に変更します。
- Redis (クラスターモードが有効) の設定
 - [Replace the only node in any Redis cluster](#) – バックアップと復元を使用して Redis クラスター内のノードを置き換える手順。
 - [Replace a replica node in any Redis cluster](#) – クラスターのダウンタイムなしでレプリカの数を増減させて Redis クラスターのリードレプリカを置き換える手順。
 - [Replace any node in a Redis \(cluster mode enabled\) shard](#) – スケールアウトおよびスケールインして Redis (クラスターモードが有効) クラスターのノードをクラスターのダウンタイムなしで置き換える動的手順。
- Redis (クラスターモードが無効) の設定

- [Replace the only node in any Redis cluster](#) – バックアップと復元を使用して Redis クラスター内の既存のノードを置き換える手順。
- [Replace a replica node in any Redis cluster](#) – クラスターのダウンタイムなしでレプリカの数を増減させて Redis クラスターのリードレプリカを置き換える手順。
- [Replace a node in a Redis \(cluster mode disabled\) cluster](#) – レプリケーションを使用して Redis (クラスターモードが無効) クラスター内のノードを置き換える手順。
- [Replace a Redis \(cluster mode disabled\) read-replica](#) – Redis (クラスターモードが無効) レプリケーショングループのリードレプリカを手動で置き換える手順です。
- [Replace a Redis \(cluster mode disabled\) primary node](#) – Redis (クラスターモードが無効) レプリケーショングループのプライマリノードを手動で置き換える手順です。

Redis ノード交換オプション

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

自動フェイルオーバーが有効になっている非クラスター構成の場合、Redis 5.0.6 以上のクラスターで置換は完全に完了しますが、クラスターは引き続きオンラインのまま、受信書き込みリクエストを処理します。Redis 4.0.10 以前の自動フェイルオーバー対応クラスターの場合、DNS 更新に関連する最大数秒の短時間の書き込み中断が発生する場合があります。

ノードが自動フェイルオーバー対応クラスターのメンバーである場合は、ElastiCache for Redis によって、パッチ適用、更新、およびその他のメンテナンス関連のノード交換時の可用性は向上します。

ElastiCache for Redis クラスタークライアントに ElastiCache を使用するように設定された構成の場合、クラスターが受信書き込みリクエストを処理している間に置換が完了するようになりました。

自動フェイルオーバーが有効になっている非クラスター構成の場合、Redis 5.0.6 以上のクラスターで置換は完全に完了しますが、クラスターは引き続きオンラインのまま、受信書き込みリクエストを処理します。Redis 4.0.10 以前の自動フェイルオーバー対応クラスターの場合、DNS 更新に関連する最大数秒の短時間の書き込み中断が発生する場合があります。

ノードがスタンダアロンである場合、Amazon ElastiCache によってまず交換ノードが起動されてから、既存のノードと同期されます。既存のノードは、この間、サービスリクエストに使用できなくなります。同期が完了すると、既存のノードが終了し、新しいノードがその代わりになります。ElastiCache は、このオペレーション中にデータを保持するために最善を尽くします。

- [メンテナンスウィンドウを変更する] – スケジュールされたメンテナンスイベントの場合、ElastiCache から E メールまたは通知イベントを受け取ります。これらの場合、スケジュールされた交換時間より前にメンテナンスウィンドウを変更すると、ノードは新しい時間に交換されます。詳細については、次を参照してください。
 - [ElastiCache クラスターの変更](#)
 - [レプリケーショングループの変更](#)

Note

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

例えば、現在が 11 月 9 日の木曜日の 15:00 で、次のメンテナンスウィンドウが 11 月 10 日金曜日の 17:00 であるとしします。以下は、3 つのシナリオとその結果です。

- メンテナンスウィンドウを金曜日の 16:00 に変更します (現在の日時以降で、次の予定メンテナンスウィンドウより前)。ノードは、11 月 10 日の金曜日の 16:00 に交換されます。
- メンテナンスウィンドウを土曜日の 16:00 に変更します (現在の日時以降で、次の予定メンテナンスウィンドウ以降)。ノードは、11 月 11 日の土曜日の 16:00 に交換されます。
- メンテナンスウィンドウを水曜日の 16:00 に変更します (同じ週内で、現在の日時より前)。ノードは、11 月 15 日の水曜日の 16:00 に交換されます。

手順については、「[メンテナンスの管理](#)」を参照してください。

- [Redis クラスターのノードのみの置き換え] – クラスターにリードレプリカがない場合は、次の手順を使ってノードを置き換えることができます。

バックアップと復元を使用してノードのみを置き換えるには

1. ノードのクラスターのスナップショットを作成します。手順については、「[手動バックアップの取得](#)」を参照してください。
 2. スナップショットからシードして、新しいクラスターを作成します。手順については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。
 3. 置き換え対象となったノードがあるクラスターを削除します。手順については、「[クラスターの削除](#)」を参照してください。
 4. アプリケーションで、古いノードのエンドポイントを新しいノードのエンドポイントに置き換えます。
- [Redis クラスターのレプリカノードの置き換え]—レプリカクラスターを置き換えるには、レプリカ数を増やします。そのためには、レプリカを追加します。その後、交換するレプリカを削除して、レプリカ数を減らします。このプロセスは動的で、クラスターのダウンタイムはありません。

Note

シャードまたはレプリケーショングループにすでに 5 つのレプリカがある場合は、ステップ 1 と 2 を逆転します。

Redis クラスターでレプリカを交換するには

1. シャードまたはレプリケーショングループにレプリカを追加してレプリカの数を増やします。詳細については、「[シャードのレプリカの数を増やす](#)」を参照してください。
 2. 置き換えるレプリカを削除します。詳細については、「[シャードのレプリカの数を減らす](#)」を参照してください。
 3. アプリケーションでエンドポイントを更新します。
- [Redis (クラスターモードが有効) シャード内のノードの置き換え]—ダウンタイムなしでクラスター内のノードを置き換えるには、オンラインリシャーディングを使用します。まずスケールアウトによりシャードを追加し、次にスケールインにより交換するノードを含むシャードを削除します。

Redis (クラスターモード有効) クラスター内のノードを置き換えるには

1. スケールアウト: 置き換えるノードがある既存のシャードと同じ設定のシャードを追加します。詳細については、「[オンラインリシャードイングによるシャードの追加](#)」を参照してください。
 2. スケールイン: 置き換えるノードがあるシャードを削除します。詳細については、「[オンラインリシャードイングによるシャードの削除](#)」を参照してください。
 3. アプリケーションでエンドポイントを更新します。
- [Redis (クラスターモードが無効) クラスター内のノードの置き換え] — クラスターがリードレプリカのない Redis (クラスターモードが無効) クラスターである場合は、以下の手順を使用してノードを置き換えます。

レプリケーションを使用してノードを交換するには (クラスターモードが無効な場合のみ)

1. プライマリとして交換対象になったノードがあるクラスターにレプリケーションを追加します。このクラスターでマルチ AZ を有効にしないでください。手順については、「[シャードがない Redis クラスターにレプリケーションを追加するには](#)」を参照してください。
 2. クラスターにリードレプリカを追加します。手順については、「[ノードをクラスターに追加するには \(コンソール\)](#)」を参照してください。
 3. 新たに作成したリードレプリカをプライマリに昇格させます。手順については、「[Redis \(クラスターモード無効\) レプリケーショングループのリードレプリカのプライマリへの昇格](#)」を参照してください。
 4. 置き換え対象となったノードを削除します。手順については、「[クラスターからのノードの削除](#)」を参照してください。
 5. アプリケーションで、古いノードのエンドポイントを新しいノードのエンドポイントに置き換えます。
- [Redis (クラスターモードが無効) リードレプリカの置き換え] – ノードがレプリケーショングループのリードレプリカである場合、ノードを交換します。

クラスターに 1 つのレプリカノードのみがあり、マルチ AZ が有効になっている場合は、マルチ AZ を無効にしてからレプリカを削除する必要があります。手順については、「[レプリケーショングループの変更](#)」を参照してください。

Redis (クラスターモードが無効) リードレプリカを置き換えるには

1. 交換対象となったレプリカを削除します。手順については、以下を参照してください。
 - [シャードのレプリカの数減らす](#)
 - [クラスターからのノードの削除](#)
 2. 置き換え対象となったレプリカと置き換える新しいレプリカを追加します。先ほど削除したレプリカと同じ名前を使用する場合は、手順 3 を省略できます。手順については、以下を参照してください。
 - [シャードのレプリカを増やす](#)
 - [Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの追加](#)
 3. アプリケーションで、古いレプリカのエンドポイントを新しいレプリカのエンドポイントに置き換えます。
 4. 開始時にマルチ AZ が無効になっている場合は、この時点で再び有効にします。手順については、「[マルチ AZ の有効化](#)」を参照してください。
- [Redis (クラスターモードが無効) プライマリノードの置き換え] – ノードがプライマリノードである場合は、まずリードレプリカをプライマリに昇格させます。次に、前はプライマリノードであったレプリカを削除します。

クラスターに 1 つのレプリカのみがあり、マルチ AZ が有効になっている場合は、マルチ AZ を無効にしてからステップ 2 のレプリカを削除する必要があります。手順については、「[レプリケーショングループの変更](#)」を参照してください。

Redis (クラスターモードが無効) プライマリノードを置き換えるには

1. リードレプリカをプライマリに昇格させます。手順については、「[Redis \(クラスターモード無効\) レプリケーショングループのリードレプリカのプライマリへの昇格](#)」を参照してください。
2. 置き換え対象となったノード (前のプライマリ) を削除します。手順については、「[クラスターからのノードの削除](#)」を参照してください。

- 置き換え対象となったレプリカと置き換える新しいレプリカを追加します。先ほど削除したノードと同じ名前を使用している場合、アプリケーションでエンドポイントの変更をスキップできます。

手順については、「[Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの追加](#)」を参照してください。

- アプリケーションで、古いノードのエンドポイントを新しいノードのエンドポイントに置き換えます。
- 開始時にマルチ AZ が無効になっている場合は、この時点で再び有効にします。手順については、「[マルチ AZ の有効化](#)」を参照してください。

ElastiCache のリザーブドノード

1 つ以上のノードを予約することは、コスト削減の手段になる場合があります。リザーブドノードには、ノードタイプと予約の期間 (1 年または 3 年) に応じて、前払い料金が請求されます。

リザーブドノードがユースケースにとってコスト削減になるかどうかを確認するには、最初に、必要なノードサイズとノード数を決定します。次に、ノードの使用量を見積もり、オンデマンドノードとリザーブドノードを使用した場合の総コストを比較します。クラスターでリザーブドノードとオンデマンドノードを組み合わせて利用することもできます。料金情報については、「[Amazon ElastiCache 料金表](#)」を参照してください。

Note

リザーブドノードは柔軟ではなく、予約したインスタンスタイプにのみ適用されます。

リザーブドノードによるコスト管理

1 つまたは複数のノードを予約すると、コスト削減の手段となる場合があります。リザーブドノードには、ノードタイプと予約の期間 (1 年または 3 年) に応じて、前払い料金が請求されます。この料金は、オンデマンドノードで発生する 1 時間あたりの使用料金よりもはるかに安くなります。

リザーブドノードがユースケースにとってコスト削減になるかどうかを確認するには、最初に、必要なノードサイズとノード数を決定します。次に、ノードの使用量を見積もり、オンデマンドノードとリザーブドノードを使用した場合の総コストを比較します。クラスターでリザーブドノードとオンデマンドノードを組み合わせて利用することもできます。料金情報については、「[Amazon ElastiCache 料金表](#)」を参照してください。

AWS リージョン、ノードタイプ、期間の長さは購入時に選択する必要があり、後で変更することはできません。

AWS Management Console、AWS CLI、または ElastiCache API を使用して、使用可能なリザーブドノードを一覧表示および購入できます。

リザーブドノードの詳細については、「[Amazon ElastiCache リザーブドノード](#)」を参照してください。

トピック

- [スタンダードリザーブドノードサービス](#)
- [従来のリザーブドノードサービス](#)

- [リザーブドノードサービスに関する詳細情報の入手](#)
- [リザーブドノードの購入](#)
- [リザーブドノードに関する詳細情報の入手](#)

スタンダードリザーブドノードサービス

Amazon ElastiCache でスタンダードリザーブドノードインスタンス (RI) を購入すると、リザーブドノードインスタンスの期間中、特定のノードインスタンスタイプと AWS リージョンに対して割引料金が適用されます。Amazon ElastiCache のリザーブドノードインスタンスを使用するには、オンデマンドインスタンスと同様に新しい ElastiCache ノードインスタンスを作成する必要があります。

新しく作成するノードインスタンスは、リザーブドノードインスタンスの仕様に完全に一致する必要があります。新しいノードインスタンスの仕様がアカウント内の既存のリザーブドノードインスタンスと一致する場合は、リザーブドノードインスタンスに適用される割引料金で請求されます。一致しない場合、ノードインスタンスはオンデマンド料金で請求されます。これらのスタンダード RI は、R5 および M5 インスタンスファミリー以降から使用可能です。

Note

次に説明する 3 つの提供タイプすべては、1 年および 3 年契約で利用できます。

提供しているタイプ

前払いなし RI は、前払い料金なしでリザーブド ElastiCache インスタンスへのアクセスを提供します。前払いなしのリザーブド ElastiCache インスタンスは、使用されたどうかにかかわらず、期間内のすべての時間は割引時間料金での請求となります。

一部前払い RI では、リザーブド ElastiCache インスタンスの一部を前払いする必要があります。期間内の残りの時間は、使用量にかかわらず、割引された時間料金で請求されます。このオプションは、次のセクションで説明する従来の重度使用オプションに代わるオプションです。

[全額前払い] RIでは、RI期間の開始時に全額の支払いが必要です。使用時間数に関係なく、残りの期間にそれ以外のコストは生じません。

従来のリザーブドノードサービス

従来のノード予約には、重度使用、中度使用、軽度使用の 3 つのレベルがあります。ノードは、どの使用量レベルでも、1 年間または 3 年間予約できます。ノードタイプ、使用量レベル、および予約

期間が合計コストに影響します。リザーブドノードを購入する前にさまざまなモデルを比較して、リザーブドノードがビジネスにもたらす節約額を確認してください。

1つの使用量レベルまたは期間で購入されたノードを、別の使用量レベルまたは期間に変換することはできません。

使用量レベル

重度使用のリザーブドノードでは、基準となる処理能力を一定に保った安定したワークロードが可能になります。重度使用のリザーブドノードの予約金は高くなりますが、インスタンスの実行時間がリザーブドノードの期間の79%を超える場合は、節約額が最も大きくなる可能性があります(最大でオンデマンド料金の70%引き)。重度使用のリザーブドノードでは、1回払いで支払います。ノードが実行されているかどうかにかかわらず、期間中は低額の使用料が時間単位で適用されます。

[中度使用のリザーブドノード]は、リザーブドノードを長い時間使用する場合に、低額の1回払いを希望するときや、ノードが停止したらすぐに支払いを中止できるようにしたいときに最適です。中度使用のリザーブドノードは、インスタンスの実行時間がリザーブドノードの期間の40%を超える予定の場合に、コスト効果の高い選択肢になります。このオプションを使用すると、オンデマンド料金から最大64%を節約できます。中度使用のリザーブドノードは、軽度使用のリザーブドノードと比べると、予約金はわずかに上回りますが、ノード実行時の時間あたりの使用料は低く抑えられます。

軽度使用のリザーブドノードは、1日に数時間、週に数日間のみ実行される定期的なワークロードに最適です。軽度使用のリザーブドノードでは、予約金を1回支払えば、ノードの実行時に時間単位で割引使用料が適用されます。ノードの実行時間がリザーブドノードの期間の17%を超えるとコスト節減が始まり、コスト節減が始まります。リザーブドノードの全期間を通してオンデマンド料金から最大56%を節約できます。

従来のリザーブドノードサービス

提供タイプ	前払いコスト	使用料	メリット
重度使用	高	時間当たりの使用料が最も低く、リザーブドノードの使用状況にかかわらず期間全体に適用されます。	リザーブドノードの実行が3年間で全体の79%を超える場合は、全体的なコストを最も抑えることができます。
中度使用	中		

提供タイプ	前払いコスト	使用料	メリット
		ノードの実行時間に 応じて使用料が時間 単位で発生します。 ノードが実行してい ないときは、時間単 位の使用量は発生し ません。	作業負荷が一定して いない場合、または 、それほど頻繁には 利用しない (3 年間で 全体の 40% を超える) 場合に適しています 。
軽度使用	低	ノードの実行時間に 応じて使用料が時間 単位で発生します。 ノードが実行してい ないときは、時間単 位の使用量は発生し ません。すべての種 類のうち最も高い料 金設定ですが、課金 されるのは、リザー ブドノードを使用し ているときに限られ ます。	常に実行することを 計画している場合は 、全体的なコストが 最も高くなります。 まれにしかリザーブ ドノードを使用しな い (3 年間で全期間の 約 15% を超える程度) 場合は、このオプシ ョンが適しています 。
オンデマンド使用 (リ ザーブドノードなし)	なし	時間単位の使用料が 最も高くなります。 ノードの実行中は常 に適用されます。	時間単位のコストが 最も高くなります。

詳細については、「[Amazon ElastiCache 料金表](#)」を参照してください。

リザーブドノードサービスに関する詳細情報の入手

リザーブドノードを購入する前に、使用可能なリザーブドノードサービスに関する情報を入手できません。

以下の例では、AWS Management Console、AWS CLI、および ElastiCache API を使用して、利用できるリザーブドノードサービスの料金表と情報を入手する方法を示します。

トピック

- [リザーブドノードサービスに関する詳細情報の入手 \(コンソール\)](#)
- [リザーブドノードサービスに関する詳細情報の入手 \(AWS CLI\)](#)
- [リザーブドノードサービスに関する詳細情報の入手 \(ElastiCache API\)](#)

リザーブドノードサービスに関する詳細情報の入手 (コンソール)

AWS Management Console を使用して利用可能なリザーブドクラスターサービスの料金とその他の情報を入手するには、次の手順に従います。

利用可能なリザーブドノードサービスの情報を入手するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[リザーブドノード] を選択します。
3. [Purchase Reserved Nodes] (リザーブドノードの購入) を選択します。
4. [Engine] (エンジン) で、Redis を選択します。
5. 利用できるサービスを確認するには、次のオプションから選択します。
 - ノードタイプ
 - 期間
 - 提供タイプ

選択後、選択したノードごとのコストと合計コストが [Reservation details] (予約の詳細) に表示されます。

6. これらのノードを購入して料金が発生することを防ぐには、[Cancel] を選択します。

リザーブドノードサービスに関する詳細情報の入手 (AWS CLI)

利用可能なリザーブドノードサービスの料金表とその他の情報を入手するには、コマンドプロンプトで次のコマンドを入力します。

```
aws elasticache describe-reserved-cache-nodes-offerings
```

このオペレーションにより、次のような出力が生成されます (JSON 形式)。

```
{
  "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
  "CacheNodeType": "cache.xxx.large",
  "Duration": 94608000,
  "FixedPrice": XXXX.X,
  "UsagePrice": X.X,
  "ProductDescription": "redis",
  "OfferingType": "All Upfront",
  "RecurringCharges": [
    {
      "RecurringChargeAmount": X.X,
      "RecurringChargeFrequency": "Hourly"
    }
  ]
},
{
  "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
  "CacheNodeType": "cache.xxx.xlarge",
  "Duration": 94608000,
  "FixedPrice": XXXX.X,
  "UsagePrice": X.X,
  "ProductDescription": "redis",
  "OfferingType": "Partial Upfront",
  "RecurringCharges": [
    {
      "RecurringChargeAmount": X.XXX,
      "RecurringChargeFrequency": "Hourly"
    }
  ]
},
{
  "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
  "CacheNodeType": "cache.xxx.large",
  "Duration": 31536000,
```

```
    "FixedPrice": X.X,  
    "UsagePrice": X.X,  
    "ProductDescription": "redis",  
    "OfferingType": "No Upfront",  
    "RecurringCharges": [  
      {  
        "RecurringChargeAmount": X.XXX,  
        "RecurringChargeFrequency": "Hourly"  
      }  
    ]  
  }  
}
```

詳細については、AWS CLI リファレンスの「[describe-reserved-cache-nodes-offerings](#)」を参照してください。

リザーブドノードサービスに関する詳細情報の入手 (ElastiCache API)

利用可能なリザーブドノードサービスの料金表と情報を入手するには、DescribeReservedCacheNodesOfferings アクションを呼び出します。

Example

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeReservedCacheNodesOfferings  
&Version=2014-12-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20141201T220302Z  
&X-Amz-Algorithm  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「[DescribeReservedCacheNodesOfferings](#)」を参照してください。

リザーブドノードの購入

以下の例では、AWS Management Console、AWS CLI、および ElastiCache API を使用してリザーブドノードサービスを購入する方法を示します。

Important

このセクションの例に従うと、AWS アカウントで料金が発生します。これを元に戻すことはできません。

トピック

- [リザーブドノードの購入 \(コンソール\)](#)
- [リザーブドノードの購入 \(AWS CLI\)](#)
- [リザーブドノードの購入 \(ElastiCache API\)](#)

リザーブドノードの購入 (コンソール)

この例では、リザーブドノード ID が myreservationID の特定のリザーブドノードサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

次の手順では、AWS Management Console を使用して、提供 ID 別にリザーブドノードサービスを購入します。

リザーブドノードを購入するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションリストで、[Reserved Nodes] (リザーブドノード) リンクを選択します。
3. [Purchase reserved nodes] (リザーブドノードを購入) ボタンを選択します。
4. [Engine] (エンジン) で、Redis を選択します。
5. 使用できるサービスを確認するには、次のオプションから選択します。
 - ノードタイプ
 - 期間
 - 提供タイプ
 - オプションの [Reserved node ID] (リザーブドノード ID)

選択後、選択したノードごとのコストと合計コストが [Reservation details] (予約の詳細) に表示されます。

6. [Purchase] (購入) を選択します。

リザーブドノードの購入 (AWS CLI)

以下の例では、リザーブドノード ID が myreservationID の特定のリザーブドクラスターサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

コマンドプロンプトで以下のコマンドを入力します。

Linux、macOS、Unix の場合:

```
aws elasticache purchase-reserved-cache-nodes-offering \  
  --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \  
  --reserved-cache-node-id myreservationID
```

Windows の場合:

```
aws elasticache purchase-reserved-cache-nodes-offering ^  
  --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^  
  --reserved-cache-node-id myreservationID
```

このコマンドにより、以下のような出力が返されます。

RESERVATION	ReservationId	Class	Start Time	Duration	
Fixed Price	Usage Price	Count	State	Description	Offering Type
RESERVATION	myreservationid	cache.xx.small	2013-12-19T00:30:23.247Z	1y	
XXX.XX USD	X.XXX USD	1	payment-pending	memcached	Medium Utilization

詳細については、AWS CLI リファレンスの「[purch-reserved-cache-nodes-offerings](#)」を参照してください。

リザーブドノードの購入 (ElastiCache API)

以下の例では、リザーブドクラスター ID が myreservationID の特定のリザーブドノードサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

以下のパラメーターを指定して `PurchaseReservedCacheNodesOffering` オペレーションを呼び出します。

- `ReservedCacheNodesOfferingId` = 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
- `ReservedCacheNodeID` = myreservationID
- `CacheNodeCount` = 1

Example

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=PurchaseReservedCacheNodesOffering  
  &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  
  &ReservedCacheNodeID=myreservationID  
  &CacheNodeCount=1  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
  &X-Amz-Date=20141201T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「[PurchaseReservedCacheNodesOffering](#)」を参照してください。

リザーブドノードに関する詳細情報の入手

購入したリザーブドノードに関する情報は、AWS Management Console、AWS CLI、および ElastiCache API を使用して入手できます。

トピック

- [リザーブドノードに関する詳細情報の入手 \(コンソール\)](#)
- [リザーブドノードに関する詳細情報の入手 \(AWS CLI\)](#)
- [リザーブドノードに関する詳細情報の入手 \(ElastiCache API\)](#)

リザーブドノードに関する詳細情報の入手 (コンソール)

次の手順では、AWS Management Console を使用して、購入したリザーブドノードに関する情報を入手します。

購入したリザーブドノードに関する情報を入手するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションリストで、[Reserved nodes] (リザーブドノード) リンクを選択します。

アカウントのリザーブドノードが [Reserved nodes] (リザーブドノード) の一覧に表示されます。リスト内のいずれかのリザーブドノードを選択して、コンソールの下部にある詳細ペインにリザーブドノードの詳細情報を表示できます。

リザーブドノードに関する詳細情報の入手 (AWS CLI)

AWS アカウントのリザーブドノードに関する情報を入手するには、コマンドプロンプトで次のコマンドを入力します。

```
aws elasticache describe-reserved-cache-nodes
```

このオペレーションにより、次のような出力が生成されます (JSON 形式)。

```
{
  "ReservedCacheNodeId": "myreservationid",
  "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
  "CacheNodeType": "cache.xx.small",
  "DataTiering": "disabled",
```



```
"Duration": "31536000",
"ProductDescription": "memcached",
"OfferingType": "Medium Utilization",
"MaxRecords": 0
}
```

詳細については、AWS CLI リファレンスの「[describe--reserved-cache-nodes](#)」を参照してください。

リザーブドノードに関する詳細情報の入手 (ElastiCache API)

AWS アカウントのリザーブドノードに関する情報を取得するには、DescribeReservedCacheNodes オペレーションを呼び出します。

Example

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReservedCacheNodes
&Version=2014-12-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「[DescribeReservedCacheNodes](#)」を参照してください。

以前の世代のノードの移行

以前の世代のノードとは、段階的に廃止されるノードタイプです。以前の世代のノードタイプを使用している既存のクラスターがない場合、ElastiCache はそのノードタイプでの新しいクラスターの作成をサポートしていません。

以前の世代のノードタイプは限られているため、クラスター内でノードが非正常になった場合、正常な置換は保証できません。このようなシナリオでは、クラスターの可用性が悪影響を受ける可能性があります。

可用性とパフォーマンスを向上させるために、クラスターを新しいノードタイプに移行することをお勧めします。移行先の推奨ノードタイプについては、「[アップグレードパス](#)」を参照してください。ElastiCache でサポートされているノードタイプと以前の世代のノードタイプの完全なリストについては、「[サポートされているノードの種類](#)」を参照してください。

Redis クラスターのノードの移行

以下の手順では、ElastiCache マネジメントコンソールを使用して、Redis クラスターのノードタイプを移行する方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。クラスターの設定によっては、次のようなダウンタイムが表示されることがあります。以下は推定値であり、特定の設定によって異なる場合があります。

- クラスターモードが無効 (シングルノード) は、主に DNS の伝播が原因で、約 60 秒表示されることがあります。
- クラスターモードが無効 (レプリカノードの場合) は、Redis 5.0.6 以降を実行しているクラスターでは約 1 秒表示されることがあります。すべての下位バージョンでは、約 10 秒かかることがあります。
- クラスターモードが有効は、約 1 秒表示されることがあります。

コンソールを使用して Redis クラスターのノードタイプを変更するには

1. コンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターのリストから、移行するクラスターを選択します。
4. アクション を選択してから、変更 を選択します。
5. ノードタイプのリストから新しいノードタイプを選択します。
6. 移行プロセスをすぐに実行する場合は、[Apply immediately] を選択します。[Apply immediately] を選択していない場合、移行プロセスはこのクラスターの次のメンテナンス期間中に実行されません。
7. [Modify] (変更) を選択します。前の手順で [Apply immediately] を選択した場合、クラスターのステータスは [modifying] に変わります。ステータスが 使用可能 に変わると、変更は完了し、新しいクラスターの使用を開始できます。

AWS CLI を使用して、Redis クラスターノードタイプを変更するには:

次に示すように [\[modify-replication-group\]](#) APIを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group /
--replication-group-id my-replication-group /
--cache-node-type new-node-type /
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^
--replication-group-id my-replication-group ^
--cache-node-type new-node-type ^
--apply-immediately
```

このシナリオでは、[\[new-node-type\]](#) の値は、移行先のノードタイプです。--apply-immediately パラメータを渡すことによって、レプリケーショングループが [変更中] から [使用可能] ステータスに変わるとすぐに適用されます。[Apply immediately] を選択していない場合、移行プロセスはこのクラスターの次のメンテナンス期間中に実行されます。

Note

InvalidCacheClusterState エラーによってクラスターを変更できない場合は、復元失敗ノードを最初に削除する必要があります。

復元失敗ノードの修正または削除

以下の手順では、復元失敗ノードを修正するか、Redis クラスターから削除する方法について説明します。ElastiCache ノードが復元失敗状態になる経緯については、「[ElastiCache ノードステータスの表示](#)」を参照してください。最初に復元失敗状態のノードをすべて削除してから、ElastiCache クラスター内の残りの旧世代のノードを新世代のノードタイプに移行し、最後に必要数のノードをあらためて追加することをお勧めします。

復元失敗ノードを削除するには (コンソール):

1. コンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。

2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターの一覧から、ノードを削除するクラスターを選択します。
4. シャードの一覧から、ノードを削除するシャードを選択します。クラスターでクラスターモードが無効の場合は、このステップをスキップします。
5. ノードのリストから、restore-failed のステータスのノードを選択します。
6. [アクション] を選択して、[ノードの削除] を選択します。

復元失敗ノードを ElastiCache クラスターから削除すると、新しい世代のタイプに移行できるようになります。詳細については、上記の「[Redis クラスターのノードの移行](#)」を参照してください。

ElastiCache クラスターにノードを追加し直す方法については、「[クラスターへのノードの追加](#)」を参照してください。

クラスターの管理

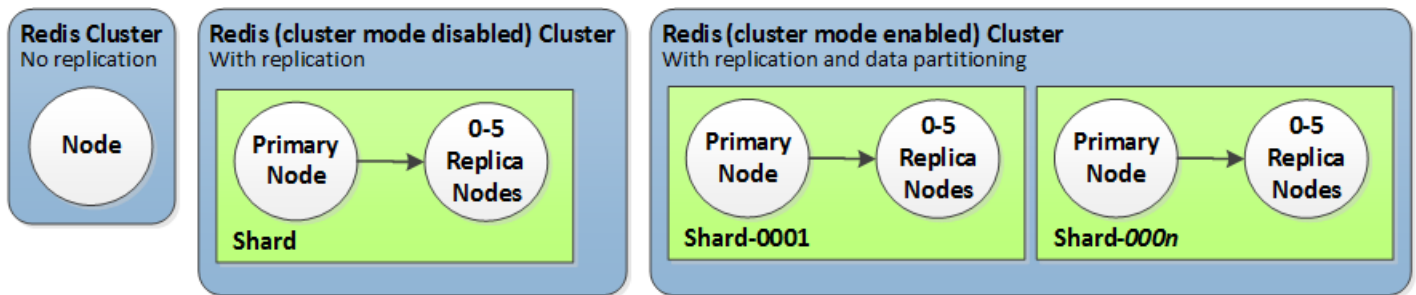
クラスターは、1 つ以上のキャッシュノードの集合であり、すべてのノードが Redis キャッシュエンジンソフトウェアのインスタンスを実行します。クラスターを作成する際に、すべてのノードで使用するエンジンとバージョンを指定します。

次の図は、一般的な Redis クラスターです。Redis クラスターは、シャード (API/CLI: ノードグループ) 内に 1 個から最大 6 個までのノードを含めることができます。単一ノードの Redis (クラスターモードが無効) クラスターはシャードを持たず、マルチノードの Redis (クラスターモードが無効) クラスターは 1 個のシャードを持ちます。Redis (クラスターモードが有効) クラスターは、最大 500 個のシャードを持つことができ、シャード間でデータを分割できます。Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

シャードに複数のノードがある場合、1つのノードは読み取り/書き込みのプライマリノードです。シャード内の他のすべてのノードは、読み取り専用のレプリカです。

一般的な Redis クラスターは次のようになります。



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

すべてのクラスターにはクラスター識別子が必要です。クラスター識別子は、お客様が指定するクラスターの名前です。この識別子は、ElastiCache API および AWS CLI コマンドを操作するとき特定のクラスターを指定します。クラスター識別子は、AWS リージョン内のお客様に対して一意である必要があります。

ElastiCache は複数のエンジンバージョンをサポートしています。特別な理由がない限り、最新バージョンを使用することをお勧めします。

ElastiCache クラスターは、Amazon EC2 インスタンスを使用してアクセスするように設計されています。Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) で起動したクラスターには、AWS の外部からアクセスできます。詳細については、「[AWS 外部からの ElastiCache リソースへのアクセス](#)」を参照してください。

サポートされている Redis バージョンのリストについては、「[サポートされている ElastiCache for Redis のバージョン](#)」を参照してください。

ネットワークタイプの選択

ElastiCache はインターネットプロトコルバージョン 4 と 6 (IPv4 と IPv6) をサポートしているため、クラスターが以下を受け入れるように設定できます。

- IPv4 接続のみ、
- IPv6 接続のみ、
- IPv4 と IPv6 の両方の接続 (デュアルスタック)

IPv6 は、[Nitro システム](#)上に構築されたすべてのインスタンスで Redis エンジンバージョン 6.2 以降を使用するワークロードでサポートされています。IPv6 を使用して ElastiCache にアクセスする場合、追加料金はかかりません。

Note

IPV6/デュアルスタックが使用可能になる前に作成されたクラスターの移行はサポートされていません。新しく作成されたクラスターのネットワークタイプの切り替えもサポートされていません。

ネットワークタイプのサブネットの設定

Amazon VPC でクラスターを作成する場合、サブネットグループを指定する必要があります。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。ElastiCache クラスターでは、デュアルスタックモードで動作するには IPv4 アドレスと IPv6 アドレスの両方が割り当てられたデュアルスタックサブネットが必要であり、IPv6 専用として動作するには IPv6 専用サブネットが必要です。

デュアルスタックの使用

クラスターモードを有効にして ElastiCache for Redis を使用する場合、アプリケーションの観点から見ると、設定エンドポイントを介してすべてのクラスターノードに接続することは、個々のキャッシュノードに直接接続することと違いがありません。これを実現するには、クラスター対応クライアントはクラスター検出プロセスを実行し、すべてのノードの設定情報をリクエストする必要があります。Redis の検出プロトコルは、ノードごとに 1 つの IP のみをサポートします。

既存のすべてのクライアントとの下位互換性を維持するために、IP 検出が導入され、これにより、検出プロトコルでアダプタイズする IP タイプ (IPv4 または IPv6 など) を選択できます。これによ

り、自動検出は 1 つの IP タイプのみに制限されますが、デュアルスタックは、ダウンタイムなしで IPv4 から IPv6 検出 IP タイプへの移行 (またはロールバック) が可能になるため、クラスターモードが有効なワークロードに引き続き有益です。

TLS が有効なデュアルスタック ElastiCache クラスター

ElastiCache クラスターで TLS が有効になっている場合、クラスター検出関数 (`cluster slots`、`cluster shards`、`cluster nodes`) は IP ではなくホスト名を返します。次に、IP の代わりにホスト名を使用して ElastiCache クラスターに接続し、TLS ハンドシェイクを実行します。つまり、クライアントは IP 検出パラメータの影響を受けません。TLS が有効なクラスターでは、IP 検出パラメータは優先 IP プロトコルに影響しません。代わりに、使用する IP プロトコルは、DNS ホスト名を解決する際にクライアントがどの IP プロトコルを使用するかによって決まります。

DNS ホスト名を解決する際に IP プロトコルプリファレンスを設定する方法の例については、[TLS 対応のデュアルスタッククラスター ElastiCache](#) を参照してください。

AWS Management Console の使用

AWS Management Console を使用してクラスターを作成する場合、[Connectivity] (接続) で、ネットワークタイプ ([IPv4]、[IPv6]、または [Dual stack] (デュアルスタック)) を選択します。Redis (クラスターモード有効) クラスターを作成していて、デュアルスタックを選択する場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

詳細については、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」または「[Redis \(クラスターモードが無効\) \(コンソール\)](#)」を参照してください。

AWS Management Console を使用してレプリケーショングループを作成する場合は、ネットワークタイプ (IPv4、IPv6、またはデュアルスタック) を選択します。デュアルスタックを選択した場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

詳細については、「[Redis \(クラスターモードが無効\) レプリケーショングループを最初から作成する](#)」または「[Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する](#)」を参照してください。

CLI の使用

CLI を使用してキャッシュクラスターを作成する場合、[create-cache-cluster](#) コマンドを使用して、`NetworkType` および `IPDiscovery` パラメータを指定します。

Linux、macOS、Unix の場合:


```
aws elasticache create-cache-cluster \  
  --cache-cluster-id "cluster-test" \  
  --engine redis \  
  --cache-node-type cache.m5.large \  
  --num-cache-nodes 1 \  
  --network-type dual_stack \  
  --ip-discovery ipv4
```

Windows の場合:

```
aws elasticache create-cache-cluster ^  
  --cache-cluster-id "cluster-test" ^  
  --engine redis ^  
  --cache-node-type cache.m5.large ^  
  --num-cache-nodes 1 ^  
  --network-type dual_stack ^  
  --ip-discovery ipv4
```

CLI を使用してクラスターモードを無効にしてレプリケーショングループを作成する場合は、[create-replication-group](#) コマンドを使用して NetworkType および IPDiscovery パラメータを指定します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id sample-repl-group \  
  --replication-group-description "demo cluster with replicas" \  
  --num-cache-clusters 3 \  
  --primary-cluster-id redis01 \  
  --network-type dual_stack \  
  --ip-discovery ipv4
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id sample-repl-group ^  
  --replication-group-description "demo cluster with replicas" ^  
  --num-cache-clusters 3 ^
```



```
--primary-cluster-id redis01 ^
--network-type dual_stack ^
--ip-discovery ipv4
```

クラスターモードを有効にしてレプリケーショングループを作成し、CLI を使用する IP 検出に IPv4 を使用する場合は、[create-replication-group](#) コマンドを使用して NetworkType および IPDiscovery パラメータを指定します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv4 \
  --region us-east-1
```

Windows の場合:

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv4 ^
  --region us-east-1
```

クラスターモードを有効にしてレプリケーショングループを作成し、CLI を使用する IP 検出に IPv6 を使用する場合は、[create-replication-group](#) コマンドを使用して NetworkType および IPDiscovery パラメータを指定します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \
```

```
--replication-group-id demo-cluster \  
--replication-group-description "demo cluster" \  
--cache-node-type cache.m5.large \  
--num-node-groups 2 \  
--engine redis \  
--cache-subnet-group-name xyz \  
--network-type dual_stack \  
--ip-discovery ipv6 \  
--region us-east-1
```

Windows の場合:

```
aws elasticache create-replication-group ^  
--replication-group-id demo-cluster ^  
--replication-group-description "demo cluster" ^  
--cache-node-type cache.m5.large ^  
--num-node-groups 2 ^  
--engine redis ^  
--cache-subnet-group-name xyz ^  
--network-type dual_stack ^  
--ip-discovery ipv6 ^  
--region us-east-1
```

データ階層化

レプリケーショングループを構成し、r6gd ファミリーのノードタイプを使用するクラスターでは、メモリとローカル SSD (ソリッドステートドライブ) ストレージの間でデータが階層化されます。データ階層化は、データをメモリに保存するだけでなく、各クラスターノードで低コストのソリッドステートドライブ (SSD) を利用することで、Redis ワークロードにコストパフォーマンスの高い新たなオプションを提供します。これは、データセット全体の最大 20% に定期的にアクセスするワークロードや、SSD 上のデータにアクセスする際に増加するレイテンシーを許容できるアプリケーションに最適です。

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

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

データ階層化は、アプリケーションワークロードへのパフォーマンスの影響を最小限に抑えるように設計されています。例えば、500 バイトの文字列値を想定した場合に、メモリ内のデータへのリクエストと比較すると、SSD に保存されたデータへのリクエストには平均で 300 マイクロ秒のレイテンシーが生じることが予想されます。

最も大きいデータ階層化ノードサイズ (cache.r6gd.16xlarge) では、単一の 500 ノードクラスターに最大 1 ペタバイトを保存できます (1 つのリードレプリカを使用する場合は 500 TB)。データ階層化は、サポートされているすべての Redis コマンドとデータ構造と互換性があります ElastiCache。この機能を使用するためのクライアント側の変更は必要ありません。

トピック

- [ベストプラクティス](#)
- [制限事項](#)
- [料金](#)
- [モニタリング](#)
- [データ階層化の使用](#)
- [データ階層化を有効にして、バックアップからクラスターにデータを復元する](#)

ベストプラクティス

推奨されるベストプラクティスを以下に示します：

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

制限事項

データ階層化には以下の制限があります。

- データ階層化は、レプリケーショングループの一部であるクラスターでのみ使用できます。

- 使用するノードタイプは、us-east-2、us-east-1、us-west-2、us-west-1、eu-west-1、eu-central-1、eu-north-1、eu-west-3、ap-northeast-1、ap-southeast-1、ap-southeast-2、ap-south-1、ca-central-1、sa-east-1 のリージョンで使用できる r6gd ファミリーのものである必要があります。
- エンジン は Redis 6.2 以降である必要があります。
- r6gd クラスターのバックアップは、r6gd を使用しなければ別のクラスターに復元できません。
- データ階層化クラスターのバックアップを Amazon S3 にエクスポートすることはできません。
- オンライン移行は、r6gd ノードタイプで実行されるクラスターではサポートされていません。
- データ階層化クラスター (r6gd ノードタイプを使用するクラスターなど) からデータ階層化を使用しないクラスター (r6g ノードタイプを使用するクラスターなど) へのスケーリングはサポートされていません。詳細については、「[Redis ElastiCache のスケーリング](#)」を参照してください。
- 自動スケーリングは、Redis バージョン 7.0.7 以降のデータ階層化を使用するクラスターでサポートされています。詳細については、「[Redis クラスター Auto ElastiCache Scaling](#)」を参照してください。
- データ階層化では、volatile-lru、allkeys-lru、volatile-lfu、allkeys-lfu、および noeviction の maxmemory ポリシーのみがサポートされます。
- 分岐なしの保存は、Redis バージョン 7.0.7 以降でサポートされています。詳細については、「[同期とバックアップの実装方法](#)」を参照してください。
- 128 MiB を超える項目は SSD に移動されません。

料金

R6gd ノードは R6g ノード (メモリのみ) と比較して 4.8 倍の合計容量 (メモリ + SSD) を備えており、最大使用率で実行すると 60 % 以上のコスト削減を実現できます。詳細については、「[の ElastiCache 料金](#)」を参照してください。

モニタリング

ElastiCache for Redis は、データ階層化を使用するパフォーマンスクラスターをモニタリングするために特別に設計されたメトリクスを提供します。SSD と比較した DRAM 内の項目の比率をモニタリングするには、「[Redis のメトリクス](#)」の CurrItems メトリクスを使用できます。パーセンテージは、(デイメンション: 階層 = メモリ * CurrItems 100) / (デイメンションフィルター CurrItems なし) として計算できます。

設定されたエビクションポリシーで許可されている場合、メモリ内の項目の割合が 5% ElastiCache を下回ると、Redis は項目の削除を開始します。削除ポリシーが設定されているノードでは、書き込みオペレーションにメモリ不足エラーが発生します。

メモリ内の項目の割合が 5% を下回る場合は、クラスターモードが有効なクラスターのスケールアウトまたはクラスターモードが無効なクラスターのスケールアップを検討することをお勧めします。スケールアップの詳細については、「」を参照してください[Redis \(クラスターモードが有効\) でのクラスターのスケールアップ](#)。データ階層化を使用する Redis クラスターのメトリクスの詳細については、「」を参照してください[Redis のメトリクス](#)。

データ階層化の使用

を使用したデータ階層化の使用 AWS Management Console

レプリケーショングループの一部としてクラスターを作成する場合は、r6gd ファミリーから cache.r6gd.xlarge などのノードタイプを選択し、データ階層化を使用します。ノードタイプを選択すると、データ階層化が自動的に有効になります。

クラスター作成の詳細については、[クラスターの作成](#) を参照してください。

を使用したデータ階層化の有効化 AWS CLI

を使用してレプリケーショングループを作成する場合 AWS CLI、cache.r6gd.xlarge などの r6gd ファミリーからノードタイプを選択し、`--data-tiering-enabled`パラメータを設定して、データ階層化を使用します。

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

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id redis-dt-cluster \  
  --replication-group-description "Redis cluster with data tiering" \  
  --num-node-groups 1 \  
  --replicas-per-node-group 1 \  
  --cache-node-type cache.r6gd.xlarge \  
  --engine redis \  
  --cache-subnet-group-name default \  
  --automatic-failover-enabled \  
  --data-tiering-enabled
```

```
--data-tiering-enabled
```

Windows の場合:

```
aws elasticache create-replication-group ^
  --replication-group-id redis-dt-cluster ^
  --replication-group-description "Redis cluster with data tiering" ^
  --num-node-groups 1 ^
  --replicas-per-node-group 1 ^
  --cache-node-type cache.r6gd.xlarge ^
  --engine redis ^
  --cache-subnet-group-name default ^
  --automatic-failover-enabled ^
  --data-tiering-enabled
```

このオペレーションを実行すると、以下のようなレスポンスが表示されます。

```
{
  "ReplicationGroup": {
    "ReplicationGroupId": "redis-dt-cluster",
    "Description": "Redis cluster with data tiering",
    "Status": "creating",
    "PendingModifiedValues": {},
    "MemberClusters": [
      "redis-dt-cluster"
    ],
    "AutomaticFailover": "enabled",
    "DataTiering": "enabled",
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:00-07:00",
    "ClusterEnabled": false,
    "CacheNodeType": "cache.r6gd.xlarge",
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}
```

データ階層化を有効にして、バックアップからクラスターにデータを復元する

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

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

データ階層化を有効にして新しいクラスターにバックアップを復元するには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで [バックアップ] を選択します。
3. バックアップのリストで、復元元のバックアップ名の左にあるチェックボックスをオンにします。
4. [復元] を選択します。
5. [クラスターの復元] ダイアログボックスに入力します。すべての [Required] (必須) フィールドと、デフォルト値から変更するその他のフィールドに入力します。
 1. [クラスター ID] – 必須。新しいクラスターの名前。
 2. クラスターモードが有効 (スケールアウト)] – クラスターの場合はこれを選択します。
 3. Node Type - cache.r6gd.xlarge または r6gd ファミリーの他のノードタイプを指定します。
 4. [シャード数] – 新しいクラスター (API/CLI: ノードグループ) に必要なシャード数を選択します。
 5. [Replicas per Shard] – 各シャードに必要なリードレプリカのノード数を選択します。
 6. [Slots and keyspaces] – シャード間でキーを分散する方法を選択します。キーの分散を指定する場合は、各シャードのキー範囲を指定するテーブルを作成します。
 7. [Availability zone(s)] – クラスターのアベイラビリティゾーンの選択方法を指定します。
 8. [Port] – 新しいクラスターで別のポートを使用する場合のみ、これを選択します。
 9. [Choose a VPC] – このクラスターを作成する VPC を選択します。
 10. [Parameter Group] – 選択したノードタイプの Redis オーバーヘッドに十分なメモリを予約するパラメータグループを選択します。
6. すべての設定が正しいことを確認したら、[作成] を選択します。

クラスター作成の詳細については、[クラスターの作成](#) を参照してください。

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

を使用してレプリケーショングループを作成する場合 AWS CLI、データ階層化は、デフォルトで cache.r6gd.xlarge などの r6gd ファミリーからノードタイプを選択し、`--data-tiering-enabled` パラメータを設定することによって使用されます。

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

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id redis-dt-cluster \  
  --replication-group-description "Redis cluster with data tiering" \  
  --num-node-groups 1 \  
  --replicas-per-node-group 1 \  
  --cache-node-type cache.r6gd.xlarge \  
  --engine redis \  
  --cache-subnet-group-name default \  
  --automatic-failover-enabled \  
  --data-tiering-enabled \  
  --snapshot-name my-snapshot
```

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group ^\  
  --replication-group-id redis-dt-cluster ^\  
  --replication-group-description "Redis cluster with data tiering" ^\  
  --num-node-groups 1 ^\  
  --replicas-per-node-group 1 ^\  
  --cache-node-type cache.r6gd.xlarge ^\  
  --engine redis ^\  
  --cache-subnet-group-name default ^\  
  --automatic-failover-enabled ^\  
  --data-tiering-enabled ^\  
  --snapshot-name my-snapshot
```

このオペレーションを実行すると、以下のようなレスポンスが表示されます。

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "redis-dt-cluster",  
    "Description": "Redis cluster with data tiering",  
    "Status": "creating",  
    "PendingModifiedValues": {},  
    "MemberClusters": [  
      "redis-dt-cluster"  
    ],  
  },  
}
```



```
    "AutomaticFailover": "enabled",
    "DataTiering": "enabled",
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:00-07:00",
    "ClusterEnabled": false,
    "CacheNodeType": "cache.r6gd.xlarge",
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}
```

クラスターを準備する

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

[AWS CloudFormation](#) を使用して ElastiCache クラスターを作成することもできます。詳細については、「AWS クラウド形成ユーザーガイド」の「[AWS::ElastiCache::CacheCluster](#)」を参照してください。これには、そのアプローチの実装方法に関するガイダンスが含まれています。

クラスターまたはレプリケーショングループを作成するときは常に、すぐにアップグレードまたは変更が必要にならないように、いくつかの準備作業をすることが推奨されます。

トピック

- [要件の特定](#)
- [ノードサイズの選択](#)

要件の特定

準備

以下の質問に対する回答を確認することで、クラスターの作成を円滑化できます。

- どのノードインスタンスタイプが必要ですか。

インスタンスのノードタイプを選択する際のガイダンスについては、「[ノードサイズの選択](#)」を参照してください。

- Amazon VPC に基づいて Virtual Private Cloud (VPC) でクラスターを起動しますか？

⚠ Important

VPC でクラスターを起動する場合、クラスターの作成を開始する前に、同じ VPC にサブネットグループを作成する必要があります。詳細については、「[サブネットおよびサブネットグループ](#)」を参照してください。

ElastiCache は、Amazon EC2 AWS を使用して 内からアクセスするように設計されています。ただし、Amazon VPC に基づく VPC で起動し、クラスターが VPC にある場合、AWS の外部アクセスを提供できません。詳細については、「[AWS 外部からの ElastiCache リソースへのアクセス](#)」を参照してください。

- パラメーター値をカスタマイズする必要がありますか。

その場合、カスタムパラメータグループを作成します。詳細については、「[パラメータグループを作成する](#)」を参照してください。

Redis を実行している場合は、reserved-memory または reserved-memory-percent を設定することを検討してください。詳細については、「[予約メモリの管理](#)」を参照してください。

- 独自の VPC セキュリティグループを作成する必要がありますか。

詳細については、[VPC のセキュリティ](#)を参照してください。

- 耐障害性をどのようにして導入しますか。

詳細については、「[障害の軽減](#)」を参照してください。

トピック

- [メモリとプロセッサの要件](#)
- [Redis Cluster の設定](#)
- [スケーリングの要件](#)
- [アクセスの要件](#)
- [リージョン、アベイラビリティゾーン、およびローカルゾーンの要件](#)

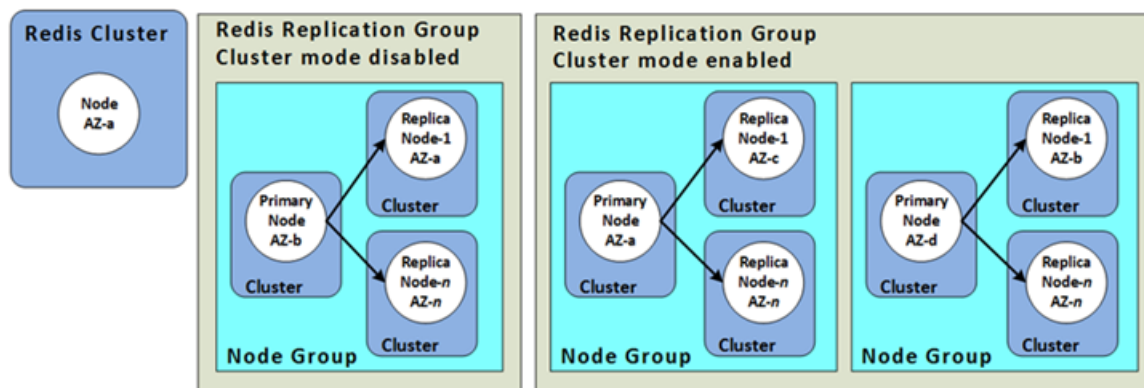
メモリとプロセッサの要件

Amazon の基本的な構成要素 ElastiCache はノードです。ノードは単体で構成される場合と、グループで構成されてクラスターを形成する場合があります。クラスターに使用するノードタイプを決定す

るときは、クラスターのノード構成および保存する必要があるデータの量を考慮する必要があります。

Redis Cluster の設定

ElastiCache for Redis クラスターは、0~500 個のシャード (ノードグループとも呼ばれます) で構成されます。Redis クラスター内のデータは、クラスターのシャード間に分割されます。アプリケーションは、エンドポイントと呼ばれるネットワークアドレスを使用して Redis クラスターに接続します。Redis のシャード内のノードは、1 つが読み取り/書き込み可能なプライマリノードの役割を持ち、他のすべては読み取り専用のセカンダリノード (リードレプリカとも呼ばれます) の役割を持ちます。ノードエンドポイントに加えて、Redis クラスター自体には設定エンドポイントと呼ばれるエンドポイントがあります。アプリケーションは、このエンドポイントを使用してクラスターに対して読み取りまたは書き込みを行うことができます。これにより、どのノードに対して読み取りまたは書き込みを行うかが決定され、for Redis ElastiCache に対して決定します。



詳細については、「[クラスターの管理](#)」を参照してください。

スケーリングの要件

すべてのクラスターは、新しい大きなノードタイプの新規クラスターを作成することでスケールアップすることができます。Redis クラスターをスケールアップする場合は、バックアップからシードすることで、新しいクラスターが空で始まるのを回避できます。

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

アクセスの要件

設計上、Amazon ElastiCache クラスターには Amazon EC2 インスタンスからアクセスします。ElastiCache クラスターへのネットワークアクセスは、クラスターを作成したアカウントに制限されます。したがって、Amazon EC2 インスタンスからクラスターに接続するには、Amazon EC2 インスタンスにクラスターへのアクセスを許可する必要があります。これを行う手順は、EC2-VPC で起動したか、または EC2-Classic で起動したかによって異なります。

クラスターを EC2-VPC で起動した場合、クラスターにネットワーク進入を許可する必要があります。EC2-Classic でクラスターを起動した場合は、インスタンスに関連付けられた Amazon Elastic Compute Cloud セキュリティグループに ElastiCache セキュリティグループへのアクセス権を付与する必要があります。詳細な手順については、このガイドの「[ステップ 3: クラスターへのアクセスの許可](#)」を参照してください。

リージョン、アベイラビリティーゾーン、およびローカルゾーンの要件

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

詳細については、次を参照してください。

- [リージョンとアベイラビリティーゾーンの選択](#)
- [ElastiCache での Local Zones の使用](#)
- [障害の軽減](#)

ノードサイズの選択

クラスターに選択するノードのサイズによって、コスト、パフォーマンス、耐障害性が変わります。

ノードサイズの選択

Graviton プロセッサの利点については、「[AWS Graviton プロセッサ](#)」を参照してください。

以下の項目に回答することで、Redis の実装に必要な最小ノードタイプを決定できます。

- 複数のクライアント接続によるスループット制限のあるワークロードを想定していますか？

この場合、Redis バージョン 5.0.6 以降を実行している場合は、Redis エンジンの代わりに利用可能な CPU をクライアント接続のオフロードに使用する AWS の拡張 I/O 機能を使用すると、スループットとレイテンシーを向上させることができます。Redis バージョン 7.0.4 以降を実行している場合、拡張 I/O に加えて、拡張 I/O 多重化によってさらに高速化されます。これは、各専用ネットワーク I/O スレッドが複数のクライアントからのコマンドを Redis エンジンにパイプラインし、コマンドをバッチ処理する Redis の機能を活用します。ElastiCache for Redis v7.1 以上では、拡張された I/O スレッド機能を拡張して、プレゼンテーション層のロジックも処理できるようにしました。プレゼンテーション層とは、クライアント入力を読み取るだけでなく、入力を Redis バイナリコマンド形式に解析する拡張 I/O スレッドを指します。これをメインスレッドに転送して

実行することで、パフォーマンスが向上します。詳細については、[ブログ投稿](#)と[サポート対象バージョン](#)のページを参照してください。

- ごく一部のデータに定期的にアクセスするワークロードがありませんか。

このような場合、Redis エンジンバージョン 6.2 以降で実行していれば、r6gd ノードタイプを選択することでデータ階層化を利用できます。データ階層化によって、最も長く使用されていないデータが SSD に保存されます。データが取得される際にレイテンシーのコストがわずかに発生しますが、コスト削減によって相殺されます。詳細については、「[データ階層化](#)」を参照してください。

詳細については、「[サポートされているノードの種類](#)」を参照してください。

- データに必要な合計メモリ量。

全般的な見積もりを得るには、キャッシュするアイテムのサイズを調べます。このサイズに、同時にキャッシュに保持するアイテムの数を掛けます。項目サイズを合理的に算出するには、まずキャッシュ項目をシリアル化して文字数をカウントします。その結果をクラスター内のシャードの数で割ります。

詳細については、「[サポートされているノードの種類](#)」を参照してください。

- 実行している Redis のバージョン。

Redis バージョン 2.8.22 より前では、フェイルオーバー、スナップショット、同期、およびレプリカをプライマリに昇格させるために、より多くのメモリを確保する必要があります。これは、十分な量のメモリを用意して、プロセスの実行時に生じるすべての書き込みに対応する必要があるためです。

Redis 2.8.22 バージョン以降では、分岐なしの保存プロセスが使用されているため、以前のプロセスよりも使用可能なメモリが少なく済みます。

詳細については、次を参照してください:

- [同期とバックアップの実装方法](#)
- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)
- アプリケーションでの書き込み負荷の大きさ。

書き込み量が多いアプリケーションでは、スナップショットの作成時またはフェイルオーバー時に、データでは使用されないより多くの使用可能メモリが必要となります。BGSAVE プロセスが実行されるたびに、BGSAVE プロセス中に発生するすべての書き込みに対応するために、データによって使用されない十分なメモリが必要です。例えば、スナップショットを作成するとき、プライ

マリクラスターをクラスター内のレプリカと同期させるとき、AOF (Append-Only File) 機能を有効にするときです。また、レプリカをプライマリに昇格させるときです (マルチ AZ が有効になっている場合)。さらに、最悪の場合、プロセス中にすべてのデータが書き換えられるときです。この場合、データのみに必要なメモリの 2 倍のサイズのノードインスタンスが必要です。

詳細については、「[Redis スナップショットを作成するのに十分なメモリがあることの確認](#)」を参照してください。

- スタンドアロンの Redis (クラスターモードが無効) クラスターを実装するか、複数のシャードを持つ Redis (クラスターモードが有効) クラスターを実装するか。

Redis (クラスターモードが無効) クラスター

Redis (クラスターモードが無効) クラスターを実装する場合は、ノードタイプがすべてのデータと前の項目で説明した必要なオーバーヘッドに対応できる必要があります。

例えば、すべてのアイテムの合計サイズを 12 GB と見積もったします。この場合、13.3 GB のメモリを搭載した `cache.m3.xlarge` ノードまたは 13.5 GB のメモリを搭載した `cache.r3.large` ノードを使用できます。ただし、BGSAVE オペレーションではより多くのメモリが必要になる場合があります。書き込み負荷の高いアプリケーションの場合、メモリ要件を少なくとも 24 GB に倍増します。したがって、27.9 GB のメモリを搭載した `cache.m3.2xlarge` または 30.5 GB のメモリを搭載した `cache.r3.xlarge` を使用します。

複数のシャードを持つ Redis (クラスターモードが有効)

複数のシャードを持つ Redis (クラスターモードが有効) クラスターを実装する場合は、ノードタイプが `bytes-for-data-and-overhead / number-of-shards` バイトのデータに対応できる必要があります。

例えば、すべてのアイテムの合計サイズが 12 GB で、シャードが 2 つあると見積もったとします。この場合、6.05 GB のメモリ (12 GB / 2) を搭載した `cache.m3.large` ノードを使用できます。ただし、BGSAVE オペレーションではより多くのメモリが必要になる場合があります。書き込み負荷の高いアプリケーションの場合、メモリ要件をシャードあたり少なくとも 12 GB に倍増します。したがって、13.3 GB のメモリを搭載した `cache.m3.xlarge` または 13.5 GB のメモリを搭載した `cache.r3.large` を使用します。

- Local Zones を使用していますか？

[\[Local Zones\]](#) を使用すると、ElastiCache クラスターなどのリソースをユーザーの近くの複数の場所に配置できます。ただし、ノードサイズを選択する場合、容量要件にかかわらず、現時点では、使用可能なノードサイズは次のサイズに制限されることに注意してください。

- 現行世代:

M5 ノードタイプ:

cache.m5.large、cache.m5.xlarge、cache.m5.2xlarge、cache.m5.4xlarge、cache.m5.

R5 ノードタイプ:

cache.r5.large、cache.r5.xlarge、cache.r5.2xlarge、cache.r5.4xlarge、cache.r5.

T3 ノードタイプ: cache.t3.micro、cache.t3.small、cache.t3.medium

キャッシュクラスターが実行中であるときに、CloudWatch に発行される、メモリの使用状況、プロセッサの使用率、キャッシュヒット、およびキャッシュミスのメトリクスをモニタリングできます。クラスターに必要なヒット率がないことや、キーが頻繁に削除されていることに気付くことがあります。これらの場合は、より高い仕様の CPU とメモリの異なるノードサイズを選択できます。

CPU 使用率をモニタリングするとき、Redis はシングルスレッドであることに注意してください。したがって、レポートされた CPU 使用率に CPU コアの数をつけて、実際の使用率を得ます。例えば、4 つのコアの CPU で使用率 20 パーセントとレポートされた場合、実際には 1 つのコアの Redis が使用率 80 パーセントで稼働しています。

クラスターの作成

次の例は、AWS Management Console、AWS CLI および ElastiCache API を使用して Redis クラスターを作成する方法を示しています。

Redis (クラスターモードが無効) (コンソール)

ElastiCache は、Redis エンジンを使用する場合にレプリケーションをサポートします。データが Redis 読み取り/書き込みプライマリクラスターに書き込まれてから読み取り専用セカンダリクラスターに伝達されるまでのレイテンシーをモニタリングするために、はクラスターに特別なキー ElastiCache を追加しますElastiCacheMasterReplicationTimestamp。このキーは、協定世界時 (UTC) の現在の時刻です。Redis クラスターが後でレプリケーショングループに追加される可能性があるため、このキーは、最初はレプリケーショングループのメンバーではない Redis クラスターであっても、すべての Redis クラスターに含まれます。レプリケーショングループの詳細については、「[レプリケーショングループを使用する高可用性](#)」を参照してください。

Redis (クラスターモードが無効) クラスターを作成するには、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」のステップに従います。

クラスターのステータスが [available] になり次第、Amazon EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、[ステップ 3: クラスターへのアクセスの許可](#)および[ステップ 4: クラスターのノードに接続する](#)を参照してください。

Important

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

Redis (クラスターモードが有効) クラスターの作成 (コンソール)

Redis 3.2.4 以降を実行している場合は、Redis (クラスターモードが有効) クラスターを作成できます。Redis (クラスターモードが有効) クラスターでは、1~500 個のシャード (API/CLI: ノードグループ) でデータを分割できます。ただし、現段階ではいくつかの制限があります。Redis (クラスターモードが無効) と Redis (クラスターモードが有効) の比較については、「[サポートされている ElastiCache for Redis のバージョン](#)」を参照してください。

ElastiCache コンソールを使用して Redis (クラスターモードが有効) クラスターを作成するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で Amazon ElastiCache コンソールを開きます。
2. 右上のリストから、このクラスターを起動する AWS リージョンを選択します。
3. ナビゲーションペインで、[Get started] (開始) を選択します。
4. [VPC の作成] を選択し、「[Virtual Private Cloud \(VPC\) の作成](#)」のステップに従います。
5. ElastiCache ダッシュボードページで、クラスターの作成を選択し、Redis クラスターの作成を選択します。
6. [クラスター設定] で、以下を実行します。
 - a. [Configure and create a new cluster] (新しいクラスターを設定および作成) を選択します。
 - b. [Cluster mode] (クラスターモード) で、[Enabled] (有効) を選択します。
 - c. [Cluster info] (クラスター情報) で、[Name] (名前) の値を入力します。
 - d. (オプション) [Description] (説明) の値を入力します。
7. [Location] (場所):

AWS Cloud

1. [AWS Cloud] (AWS クラウド) の場合、[Multi-AZ] (マルチ AZ) および [Auto-failover] (自動フェイルオーバー) のデフォルト設定を受け入れることをお勧めします。詳細については、「[マルチ AZ を使用した for Redis のダウンタイムの最小化 ElastiCache](#)」を参照してください。
2. [Cluster settings] (クラスター設定)
 - a. [Engine version] (エンジンバージョン) で、使用可能なバージョンを選択します。
 - b. [Port] (ポート) で、デフォルトポート 6379 を使用します。異なるポートを使用する理由がある場合は、そのポート番号を入力します。
 - c. [パラメータグループ] で、パラメータグループを選択するか、新しいパラメータグループを作成します。パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「[Redis 固有のパラメータ](#)」および「[パラメータグループを作成する](#)」を参照してください。

Note

パラメータグループを選択してエンジン設定値を設定すると、そのパラメータグループが Global Datastore 内のすべてのクラスターに適用されます。[パラメータグループ] ページの yes/no [グローバル] 属性は、パラメータグループがグローバルデータストアの一部であるかどうかを示します。

- d. [ノードタイプ] で、下向き矢印



を選択します。[ノードタイプの変更] ダイアログボックスで、必要なノードタイプの [インスタンスファミリー] の値を選択します。次に、このクラスターで使用するノードタイプを選択し、[保存] を選択します。

詳細については、「[ノードサイズを選択](#)」を参照してください。

r6gd ノードタイプを選択すると、データ階層化が自動的に有効になります。詳細については、「[データ階層化](#)」を参照してください。

- e. [シャード数] で、この Redis (クラスターモードが有効) クラスターに必要なシャード (パーティション/ノードグループ) の数を選択します。

Redis (クラスターモードが有効) の一部のバージョンでは、クラスター内のシャード数を動的に変更できます。

- [Redis 3.2.10 以降] – クラスターで Redis 3.2.10 以降のバージョンを実行している場合は、クラスター内のシャード数を動的に変更できます。詳細については、「[Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)」を参照してください。
- [その他の Redis バージョン] – クラスターでバージョン 3.2.10 より前のバージョンの Redis を実行している場合は、別の方法があります。この場合、クラスター内のシャード数を変更するには、新しいシャード数を使用して新しいクラスターを作成します。詳細については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

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

Redis (クラスターモードが有効) には、次の制限があります。

- マルチ AZ が有効になっている場合は、シャードごとに少なくとも 1 つのレプリカがあることを確認してください。
- コンソールを使用してクラスターを作成する場合、シャードごとのレプリカ数は同じになります。
- シャードあたりのリードレプリカ数は固定され、変更できません。シャード (API/ CLI: ノードグループ) あたりのレプリカ数を増減する必要がある場合は、新しいレプリカ数で新しいクラスターを作成する必要があります。詳細については、[「外部で作成されたバックアップによる新しい独自設計型クラスターのシード」](#)を参照してください。

3. [Connectivity] (接続) で

- a. [Network type] (ネットワークタイプ) で、このクラスターがサポートする IP バージョンを選択します。
- b. サブネットグループで、このクラスターに適用するサブネットを選択します。はそのサブネットグループ ElastiCache を使用して、ノードに関連付けるサブネットと IP アドレスを選択します。ElastiCache クラスターには、デュアルスタックモードで動作するように割り当てられた IPv4 アドレスと IPv6 アドレスの両方を持つデュアルスタックサブネットと、IPv6-onlyとして動作するように IPv6-onlyサブネットが必要です。

新しいサブネットグループを作成するときは、そのサブネットグループが属する VPC ID を入力します。

[Discovery IP type] (検出 IP タイプ) を選択します。選択したプロトコルの IP アドレスのみが返されます。

詳細については、以下を参照してください。

- [ネットワークタイプの選択](#).
- [VPC にサブネットを作成します](#)。

[ElastiCache での Local Zones の使用](#) である場合は、ローカルゾーンにあるサブネットを作成または選択する必要があります。


詳細については、「[サブネットおよびサブネットグループ](#)」を参照してください。

4. [Availability zone placements] (アベイラビリティゾーンの配置) には 2 つのオプションがあります。
 - 設定なし — ElastiCache アベイラビリティゾーンを選択します。
 - [アベイラビリティゾーンの指定] – 各クラスターに対するアベイラビリティゾーンを指定します。

アベイラビリティゾーンの指定を選択した場合、クラスターのシャードごとにリストからアベイラビリティゾーンを選択します。

詳細については、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

5. [Next] (次へ) を選択します。
6. [Advanced Redis settings] (Redis の詳細設定) で
 - [Security] (セキュリティ):
 - i. データを暗号化するには、次のオプションがあります。
 - 保管時の暗号化 – ディスクに保存されているデータの暗号化を有効にします。詳細については、「[保管時の暗号化](#)」を参照してください。

 Note

カスタマーマネージド AWS KMS キーを選択し、キーを選択することで、別の暗号化キーを指定することもできます。詳細については、「[AWS KMS のカスタマー管理の CMK の使用](#)」を参照してください。

- [転送中の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「[転送中の暗号化](#)」を参照してください。Redis エンジンバージョン 6.0 以降では、転送中の暗号化を有効にすると、次のアクセスコントロールオプションのいずれかを指定するよう求められます。
 - アクセスコントロールなし — これがデフォルトの設定です。これは、クラスターへのユーザーアクセスに制限がないことを示します。
 - [ユーザーグループのアクセスコントロールリスト] — クラスターにアクセスできるユーザーのセットが定義されているユーザーグループを選択しま

す。詳細については、「[コンソールおよび CLI を使用したユーザーグループの管理](#)」を参照してください。

- [Redis AUTH デフォルトユーザー] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。
- [Redis AUTH] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。

Note

3.2.6 以降の Redis バージョン (バージョン 3.2.10 を除く) では、Redis AUTH のみがオプションとなります。

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

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

7. 自動バックアップを定期的にスケジュールする場合は、[自動バックアップの有効化] を選択し、自動バックアップを保持して自動的に削除するまでの日数を入力します。自動バックアップを定期的にスケジュールしない場合は、[自動バックアップを有効化] チェックボックスをオフにします。いずれの場合も、常に手動バックアップを作成するオプションがあります。

Redis のバックアップと復元の詳細については、「[スナップショットおよび復元](#)」を参照してください。

8. (オプション) メンテナンスウィンドウを指定します。[メンテナンスウィンドウ] は、ElastiCache がクラスターのシステムメンテナンスを毎週スケジュールする時間の長さ (通常は 1 時間単位) です。ElastiCache がメンテナンスの日時を選択することを許可するか ([No preference])、自分で日時と期間を選択できます ([Specify maintenance window])。メンテナンスウィンドウを指定を選択した場合は、リストからメンテナンス期間の Start day、開始時間および期間を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「[メンテナンスの管理](#)」を参照してください。

9. (オプション) [ログ]:

- [ログの形式] の下で、[テキスト] または [JSON] を選択します。
- 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
- ログ送信先 で、新規作成 を選択し、CloudWatch ログロググループ名または Firehose ストリーム名を入力するか、既存選択 を選択し、CloudWatch ログロググループ名または Firehose ストリーム名を選択します。

10. タグ では、クラスターやその他の ElastiCache リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。詳細については、「[ElastiCache リソースのタグ付け](#)」を参照してください。

11. [次へ] をクリックします。

12. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、[Create] (作成) を選択します。

On premises

1. [On premises] (オンプレミス) では、[Auto-failover] (自動フェイルオーバー) を有効のままにしておくことをお勧めします。詳細については、「[マルチ AZ を使用した for Redis のダウンタイムの最小化 ElastiCache](#)」を参照してください。
2. 「[Outposts の使用](#)」のステップに従います。

ElastiCache API または ElastiCache コンソール AWS CLI の代わりに同等の を作成するには、以下を参照してください。

- API: [CreateReplicationGroup](#)
- CLI: [create-replication-group](#)

クラスターのステータスが 使用可能 になり次第、EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、「[ステップ 3: クラスターへのアクセスの許可](#)および[ステップ 4: クラスターのノードに接続する](#)」を参照してください。

⚠ Important

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

クラスターの作成 (AWS CLI)

を使用してクラスターを作成するには AWS CLI、`create-cache-cluster` コマンドを使用します。

Important

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

Redis (クラスターモードが無効) クラスターの作成 (CLI)

Example — リードレプリカのない Redis (クラスターモードが無効) クラスター

次の CLI コードでは、レプリカのない Redis (クラスターモードが無効) キャッシュクラスターを作成します。

Note

r6gd ファミリーのノードタイプを使用してクラスターを作成する場合は、`data-tiering-enabled` パラメータを渡す必要があります。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-cluster \  
--cache-cluster-id my-cluster \  
--cache-node-type cache.r4.large \  
--engine redis \  
--num-cache-nodes 1 \  
--cache-parameter-group default.redis6.x \  
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Windows の場合:

```
aws elasticache create-cache-cluster ^  
--cache-cluster-id my-cluster ^  
--cache-node-type cache.r4.large ^  
--engine redis ^
```



```
--num-cache-nodes 1 ^  
--cache-parameter-group default.redis6.x ^  
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Redis (クラスターモードが有効) クラスターの作成 (AWS CLI)

Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ) は、`create-cache-cluster` オペレーションを使用して作成できません。Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ) を作成するには、「[Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)」を参照してください。

詳細については、AWS CLI 「」の ElastiCache リファレンストピックを参照してください [create-replication-group](#)。

クラスターの作成 (ElastiCache API)

ElastiCache API を使用してクラスターを作成するには、`CreateCacheCluster` アクションを使用します。

Important

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

トピック

- [Redis \(クラスターモードが無効\) キャッシュクラスターの作成 \(ElastiCache API\)](#)
- [Redis \(クラスターモードが有効\) でのキャッシュクラスターの作成 \(ElastiCache API\)](#)

Redis (クラスターモードが無効) キャッシュクラスターの作成 (ElastiCache API)

次のコードは、Redis (クラスターモードが無効) キャッシュクラスター (ElastiCache API) を作成します。

改行は読みやすくするために追加しています。

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=CreateCacheCluster
&CacheClusterId=my-cluster
&CacheNodeType=cache.r4.large
&CacheParameterGroup=default.redis3.2
&Engine=redis
&EngineVersion=3.2.4
&NumCacheNodes=1
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
&Timestamp=20150508T220302Z
&Version=2015-02-02
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20150508T220302Z
&X-Amz-Expires=20150508T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Signature=<signature>
```

Redis (クラスターモードが有効) でのキャッシュクラスターの作成 (ElastiCache API)

Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ)

は、CreateCacheCluster オペレーションを使用して作成できません。Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ) を作成するには、「[Redis \(クラスターモードが有効\) でのレプリケーショングループを最初から作成する \(ElastiCache API\)](#)」を参照してください。

詳細については、ElastiCache API リファレンストピック「」を参照してください [CreateReplicationGroup](#)。

クラスターの詳細を表示する

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

Redis (クラスターモードが無効) クラスターの詳細の表示 (コンソール)

ElastiCache コンソール、AWS CLI for ElastiCache、または ElastiCache API を使用して、Redis (クラスターモードが無効) クラスターの詳細を表示できます。

次の手順は、ElastiCache コンソールを使用して Redis (クラスターモードが無効) クラスターの詳細を表示する方法を示しています。

Redis (クラスターモードが無効) クラスターの詳細を表示するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ElastiCache コンソールのダッシュボードで [Redis] を選択し、いずれかのバージョンの Redis を実行しているすべてのクラスターを一覧表示します。
3. クラスターの詳細を表示するには、クラスター名の左側にあるボックスを選択します。クラスター化された Redis ではなく、Redis エンジンを実行しているクラスターを必ず選択してください。これにより、クラスターのプライマリエンドポイントを含む、クラスターの詳細が表示されます。
4. ノード情報を表示するには
 - a. クラスターの名前を選択します。
 - b. [Shards and nodes] (シャードとノード) タブを選択します。これにより、クラスターから読み込むために使用する必要があるノードのエンドポイントを含む、各ノードの詳細が表示されます。
5. メトリクスを表示するには、[Metrics] (メトリクス) タブを選択します。このタブには、クラスター内のすべてのノードに関連するメトリクスが表示されます。詳細については、「[CloudWatch メトリクスの使用状況のモニタリング](#)」を参照してください。
6. ログを表示するには、[Logs] (ログ) タブを選択します。このタブには、クラスターがスローログとエンジンログのどちらを使用しているかが示され、関連する詳細が表示されます。詳細については、「[ログ配信](#)」を参照してください。
7. [Network and security] (ネットワークとセキュリティ) タブを選択すると、クラスターのネットワーク接続とサブネットグループ設定の詳細が表示されます。詳細については、「[サブネットおよびサブネットグループ](#)」を参照してください。

8. [Maintenance] (メンテナンス) タブを選択すると、クラスターのメンテナンス設定の詳細が表示されます。詳細については、「[メンテナンスの管理](#)」を参照してください。
9. [Service updates] (サービスの更新) タブを選択すると、利用可能なサービスの更新の詳細と推奨適用期限が表示されます。詳細については、「[でのサービスの更新 ElastiCache](#)」を参照してください。
10. [Tags] (タグ) タブを選択すると、クラスターリソースに適用されているタグの詳細が表示されます。詳細については、「[ElastiCache リソースのタグ付け](#)」を参照してください。

Redis (クラスターモードが有効) クラスターの詳細の表示 (コンソール)

ElastiCache コンソール、AWS CLI for ElastiCache、または ElastiCache API を使用して、Redis (クラスターモードが有効) クラスターの詳細を表示できます。

次の手順は、ElastiCache コンソールを使用して Redis (クラスターモードが有効) クラスターの詳細を表示する方法を示しています。

Redis (クラスターモードが有効) クラスターの詳細を表示するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 右上隅にある一覧から、目的の AWS リージョンを選択します。
3. ElastiCache コンソールのダッシュボードで [Redis] を選択し、いずれかのバージョンの Redis を実行しているすべてのクラスターを一覧表示します。
4. Redis (クラスターモードが有効) クラスターの詳細を表示するには、クラスター名の左側にあるボックスを選択します。単なる Redis ではなく、Clustered Redis エンジンを実行しているクラスターを必ず選択してください。

クラスターの下画面が展開され、クラスターに関する詳細 (クラスターの設定エンドポイントなど) が表示されます。

5. クラスター内のシャード数とシャードごとのノード数を一覧表示するには、[Shards and nodes] (シャードとノード) タブを選択します。
6. ノード固有の情報を表示するには
 - シャードの ID を選択します。

これにより、クラスターからデータを読み取るために必要な各ノードのエンドポイントなどの情報がノードごとに表示されます。

7. メトリクスを表示するには、[Metrics] (メトリクス) タブを選択します。このタブには、クラスター内のすべてのノードに関連するメトリクスが表示されます。詳細については、「[CloudWatch メトリクスの使用状況のモニタリング](#)」を参照してください。
8. ログを表示するには、[Logs] (ログ) タブを選択します。このタブには、クラスターがスローログとエンジンログのどちらを使用しているかが示され、関連する詳細が表示されます。詳細については、「[ログ配信](#)」を参照してください。
9. [Network and security] (ネットワークとセキュリティ) タブを選択すると、クラスターのネットワーク接続とサブネットグループ設定、使用されている場合は、クラスターで有効にされている暗号化の方法の詳細が表示されます。詳細については、[サブネットおよびサブネットグループ](#) および [Amazon ElastiCache のデータセキュリティ](#) を参照してください。
10. [Maintenance] (メンテナンス) タブを選択すると、クラスターのメンテナンス設定の詳細が表示されます。詳細については、「[メンテナンスの管理](#)」を参照してください。
11. [Service updates] (サービスの更新) タブを選択すると、利用可能なサービスの更新の詳細と推奨適用期限が表示されます。詳細については、「[でのサービスの更新 ElastiCache](#)」を参照してください。
12. [Tags] (タグ) タブを選択すると、クラスターリソースに適用されているタグの詳細が表示されます。詳細については、「[ElastiCache リソースのタグ付け](#)」を参照してください。

クラスターの詳細の表示 (AWS CLI)

次のコードは、*my-cluster* の詳細をリストします。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

create-cache-cluster コマンドを使用してクラスターが 1 つのキャッシュノードと 0 シャードで作成された場合は、*my-cluster* を自分のクラスター名に置き換えます。

```
{
  "CacheClusters": [
    {
      "CacheClusterStatus": "available",
      "SecurityGroups": [
        {
          "Status": "active",
          "SecurityGroupId": "sg-dbe93fa2"
        }
      ]
    }
  ],
}
```

```

    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "wed:12:00-wed:13:00",
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "08:30-09:30",
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false,
    "CacheClusterId": "my-cluster1",
    "CacheClusterCreateTime": "2018-02-26T21:06:43.420Z",
    "PreferredAvailabilityZone": "us-west-2c",
    "AuthTokenEnabled": false,
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "ParameterApplyStatus": "in-sync",
        "CacheParameterGroupName": "default.redis3.2"
    },
    "SnapshotRetentionLimit": 0,
    "AutoMinorVersionUpgrade": true,
    "EngineVersion": "3.2.10",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
}

```

```

{
  "CacheClusters": [
    {
      "SecurityGroups": [
        {
          "Status": "active",
          "SecurityGroupId": "sg-dbe93fa2"
        }
      ],
      "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
      "AuthTokenEnabled": false,
      "CacheSubnetGroupName": "default",
      "SnapshotWindow": "12:30-13:30",
      "AutoMinorVersionUpgrade": true,
      "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",

```

```
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": false,
    "PreferredAvailabilityZone": "us-west-2a",
    "TransitEncryptionEnabled": false,
    "ReplicationGroupId": "my-cluster2",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
    "CacheClusterId": "my-cluster2-001",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis6.x"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "6.0",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  },
  {
    "SecurityGroups": [
      {
        "Status": "active",
        "SecurityGroupId": "sg-dbe93fa2"
      }
    ],
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
    "AuthTokenEnabled": false,
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": false,
    "PreferredAvailabilityZone": "us-west-2b",
    "TransitEncryptionEnabled": false,
    "ReplicationGroupId": "my-cluster2",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
    "CacheClusterId": "my-cluster2-002",
    "PendingModifiedValues": {},
```

```

    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis6.x"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "6.0",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  },
  {
    "SecurityGroups": [
      {
        "Status": "active",
        "SecurityGroupId": "sg-dbe93fa2"
      }
    ],
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
    "AuthTokenEnabled": false,
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": false,
    "PreferredAvailabilityZone": "us-west-2c",
    "TransitEncryptionEnabled": false,
    "ReplicationGroupId": "my-cluster2",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
    "CacheClusterId": "my-cluster2-003",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis3.2"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "3.2.10",

```



```

    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  }

```

```

{
  "CacheClusters": [
    {
      "SecurityGroups": [
        {
          "Status": "active",
          "SecurityGroupId": "sg-dbe93fa2"
        }
      ],
      "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
      "AuthTokenEnabled": true,
      "CacheSubnetGroupName": "default",
      "SnapshotWindow": "12:30-13:30",
      "AutoMinorVersionUpgrade": true,
      "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
      "CacheClusterStatus": "available",
      "AtRestEncryptionEnabled": true,
      "PreferredAvailabilityZone": "us-west-2a",
      "TransitEncryptionEnabled": true,
      "ReplicationGroupId": "my-cluster3",
      "Engine": "redis",
      "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
      "CacheClusterId": "my-cluster3-0001-001",
      "PendingModifiedValues": {},
      "CacheNodeType": "cache.r4.large",
      "DataTiering": "disabled",
      "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "ParameterApplyStatus": "in-sync",
        "CacheParameterGroupName": "default.redis6.x.cluster.on"
      },
      "SnapshotRetentionLimit": 0,
      "EngineVersion": "6.0",
      "CacheSecurityGroups": [],
      "NumCacheNodes": 1
    },
    {
      "SecurityGroups": [

```

```
    {
      "Status": "active",
      "SecurityGroupId": "sg-dbe93fa2"
    }
  ],
  "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
  "AuthTokenEnabled": true,
  "CacheSubnetGroupName": "default",
  "SnapshotWindow": "12:30-13:30",
  "AutoMinorVersionUpgrade": true,
  "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
  "CacheClusterStatus": "available",
  "AtRestEncryptionEnabled": true,
  "PreferredAvailabilityZone": "us-west-2b",
  "TransitEncryptionEnabled": true,
  "ReplicationGroupId": "my-cluster3",
  "Engine": "redis",
  "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
  "CacheClusterId": "my-cluster3-0001-002",
  "PendingModifiedValues": {},
  "CacheNodeType": "cache.r4.large",
  "DataTiering": "disabled",
  "CacheParameterGroup": {
    "CacheNodeIdsToReboot": [],
    "ParameterApplyStatus": "in-sync",
    "CacheParameterGroupName": "default.redis3.2.cluster.on"
  },
  "SnapshotRetentionLimit": 0,
  "EngineVersion": "3.2.6",
  "CacheSecurityGroups": [],
  "NumCacheNodes": 1
},
{
  "SecurityGroups": [
    {
      "Status": "active",
      "SecurityGroupId": "sg-dbe93fa2"
    }
  ],
  "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
  "AuthTokenEnabled": true,
  "CacheSubnetGroupName": "default",
```

```

    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": true,
    "PreferredAvailabilityZone": "us-west-2c",
    "TransitEncryptionEnabled": true,
    "ReplicationGroupId": "my-cluster3",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
    "CacheClusterId": "my-cluster3-0001-003",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis6.x.cluster.on"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "6.0",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  },
  {
    "SecurityGroups": [
      {
        "Status": "active",
        "SecurityGroupId": "sg-dbe93fa2"
      }
    ],
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
    "AuthTokenEnabled": true,
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": true,
    "PreferredAvailabilityZone": "us-west-2b",
    "TransitEncryptionEnabled": true,
    "ReplicationGroupId": "my-cluster3",
    "Engine": "redis",

```

```

    "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
    "CacheClusterId": "my-cluster3-0002-001",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis6.x.cluster.on"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "6.0",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  },
  {
    "SecurityGroups": [
      {
        "Status": "active",
        "SecurityGroupId": "sg-dbe93fa2"
      }
    ],
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
    "AuthTokenEnabled": true,
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": true,
    "PreferredAvailabilityZone": "us-west-2c",
    "TransitEncryptionEnabled": true,
    "ReplicationGroupId": "my-cluster3",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
    "CacheClusterId": "my-cluster3-0002-002",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis3.2.cluster.on"
    }
  }

```

```
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "3.2.6",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  },
  {
    "SecurityGroups": [
      {
        "Status": "active",
        "SecurityGroupId": "sg-dbe93fa2"
      }
    ],
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
    "AuthTokenEnabled": true,
    "CacheSubnetGroupName": "default",
    "SnapshotWindow": "12:30-13:30",
    "AutoMinorVersionUpgrade": true,
    "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
    "CacheClusterStatus": "available",
    "AtRestEncryptionEnabled": true,
    "PreferredAvailabilityZone": "us-west-2a",
    "TransitEncryptionEnabled": true,
    "ReplicationGroupId": "my-cluster3",
    "Engine": "redis",
    "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
    "CacheClusterId": "my-cluster3-0002-003",
    "PendingModifiedValues": {},
    "CacheNodeType": "cache.r4.large",
    "DataTiering": "disabled",
    "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "ParameterApplyStatus": "in-sync",
      "CacheParameterGroupName": "default.redis6.x.cluster.on"
    },
    "SnapshotRetentionLimit": 0,
    "EngineVersion": "6.0",
    "CacheSecurityGroups": [],
    "NumCacheNodes": 1
  }
]
}
```

AWS Management Console を使用してクラスターを作成する場合(クラスターノードが 1 つ以上のシャードで有効または無効)、以下のコマンドを使用してクラスターの詳細を記述します (*my-cluster* をレプリケーショングループの名前 (自分のクラスターの名前) に置き換えます)。

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

詳細については、「AWS CLI for ElastiCache トピック [describe-cache-clusters](#) の」を参照してください。

クラスターの詳細の表示 (ElastiCache API)

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

次のコードは my-cluster の詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterId=my-cluster  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&MaxRecords=25  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、ElastiCache API リファレンストピック「[DescribeCacheClusters](#)」を参照してください。

ElastiCache クラスターの変更

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

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

クラスターのパラメータを変更すると、変更はすぐにクラスターに適用されるか、クラスターが再起動されてから適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。特定のパラメータの変更が適用されるタイミングを確認するには、[Redis 固有のパラメータ](#) のテーブルの「詳細」列の「変更が有効になる」セクションを参照してください。

の使用 AWS Management Console

クラスターを変更するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. 右上隅のリストから、変更するクラスターがある AWS リージョンを選択します。
3. ナビゲーションペインで、変更するクラスターで実行されているエンジンを選択します。

選択したエンジンのクラスターが一覧表示されます。

4. クラスターのリストで、変更するクラスターの名前を選択します。
5. アクション を選択してから、変更 を選択します。

[Modify Cluster] ウィンドウが表示されます。

6. [クラスターの変更] ウィンドウで、必要な変更を行います。オプションには以下が含まれます。
 - 説明
 - クラスターモード - クラスターモードを [無効] から [有効] に変更するには、まずクラスターモードを [互換] に設定する必要があります。

互換モードでは、Redis クライアントはクラスターモード有効およびクラスターモード無効の両方を使用して接続することができます。すべての Redis クライアントを移行してクラスターモードを有効にしたら、クラスターモードの設定を完了し、クラスターモードを [有効] に設定できます。

- エンジンバージョンの互換性

⚠ Important

最新のエンジンバージョンにアップグレードできます。例えば、5.0.6 から 6.0 へのメジャーエンジンのバージョンをアップグレードする場合は、新しいエンジンバージョンと互換性があるパラメータグループファミリーを選択する必要があります。その設定方法の詳細については、「[エンジンバージョンとアップグレード](#)」を参照してください。ただし、既存のクラスターを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。

- VPC セキュリティグループ
- パラメータグループ
- ノードの種類

i Note

クラスターが r6gd ファミリーのノードタイプを使用している場合は、そのファミリー内からのみ別のノードサイズを選択できます。r6gd ファミリーからノードタイプを選択すると、データ階層化が自動的に有効になります。詳細については、[データ階層化](#)を参照してください。

- マルチ AZ
- 自動フェイルオーバー (クラスターモードが無効のみ)
- 自動バックアップの有効化
- バックアップノード ID
- バックアップの保持期間
- バックアップウィンドウ
- SNS 通知のトピック

[Apply Immediately (すぐに適用)] チェックボックスはエンジンバージョンの変更にも適用されます。変更をすぐに適用するには、[Apply Immediately (すぐに適用)] チェックボックスをオンにします。このチェックボックスがオフになっている場合、ノードタイプとエンジンバージョンの変更は次回のメンテナンスウィンドウ中に適用されます。その他の変更 (メンテナンス期間の変更など) はすぐに適用されます。

7. [変更] を選択します。

ログの配信を有効または無効にするには

1. クラスターのリストから、変更するクラスターを選択します。横にあるチェックボックスではなく、[クラスター名] を選択します。
2. [Cluster details] (クラスターの詳細) ページで、[Logs] (ログ) タブを選択します。
3. スローログを有効/無効にするには、[Enable] (有効化) または [Disable] (無効化) を選択します。

有効化を選択した場合:

- a. [Log format] (ログの形式) の下で、[JSON] または [Text] (テキスト) を選択します。
- b. ログ送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
- c. ログ送信先 で、新規作成 を選択し、 CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を入力します。または、既存の を選択 を選択し、 CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を選択します。
- d. [Enable (有効化)] を選択します。

設定を変更するには:

1. [Modify] (変更) を選択します。
2. [Log format] (ログの形式) の下で、[JSON] または [Text] (テキスト) を選択します。
3. 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
4. ログ送信先 で、新規作成 を選択し、 CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を入力します。または、既存の を選択 を選択し、 CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を選択します。

の使用 AWS CLI

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

⚠ Important

最新のエンジンバージョンにアップグレードできます。例えば、5.0.6 から 6.0 へのメジャーエンジンのバージョンをアップグレードする場合は、新しいエンジンバージョンと互換性があるパラメータグループファミリーを選択する必要があります。その設定方法の詳細については、「[エンジンバージョンとアップグレード](#)」を参照してください。ただし、既存のクラスターを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-cluster \  
  --preferred-maintenance-window sun:23:00-mon:02:00
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^  
  --cache-cluster-id my-cluster ^  
  --preferred-maintenance-window sun:23:00-mon:02:00
```

--apply-immediately パラメータは、ノードタイプとエンジンバージョンの変更、クラスターのノード数の変更にも適用されます。これらの変更のいずれもすぐに適用する場合は、--apply-immediately パラメータを使用します。これらの変更を次のメンテナンス期間に延期する場合は、--no-apply-immediately パラメータを使用します。その他の変更 (メンテナンス期間の変更など) はすぐに適用されます。

詳細については、「[の ElastiCache トピック AWS CLI](#)」を参照してください [modify-cache-cluster](#)。

ElastiCache API の使用

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

⚠ Important

最新のエンジンバージョンにアップグレードできます。例えば、5.0.6 から 6.0 へのメジャーエンジンのバージョンをアップグレードする場合は、新しいエンジンバージョンと互換性があるパラメータグループファミリーを選択する必要があります。その設定方法の詳細については、「[エンジンバージョンとアップグレード](#)」を参照してください。ただし、既存のクラスターを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。

改行は読みやすくするために追加しています。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyCacheCluster  
&CacheClusterId=my-cluster  
&PreferredMaintenanceWindow=sun:23:00-mon:02:00  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150901T220302Z  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Date=20150202T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20150901T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

ApplyImmediately パラメータは、ノードタイプとエンジンバージョンの変更、クラスターのノード数の変更にも適用されます。これらの変更のいずれもすぐに適用する場合は、ApplyImmediately パラメータを true に設定します。これらの変更を次のメンテナンス期間に延期する場合は、ApplyImmediately パラメータを false に設定します。その他の変更(メンテナンス期間の変更など)はすぐに適用されます。

詳細については、ElastiCache 「API リファレンストピック」を参照してください [ModifyCacheCluster](#)。

クラスターへのノードの追加

Redis (クラスターモードが有効) クラスターを再設定するには、「[Redis \(クラスターモードが有効\)でのクラスターのスケールリング](#)」を参照してください。

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

AWS Management Console を使用する場合

単一ノード Redis (クラスターモードが無効) クラスター (レプリケーションが有効でないクラスター) にノードを追加する場合は、2 ステッププロセスとなり、最初のステップでレプリケーションを追加し、次のステップでレプリカノードを追加します。

トピック

- [シャードがない Redis クラスターにレプリケーションを追加するには](#)
- [ノードをクラスターに追加するには \(コンソール\)](#)

次の手順では、レプリケーションが有効でない単一ノード Redis にレプリケーションを追加します。レプリケーションを追加すると、既存のノードはレプリケーションが有効なクラスターのプライマリノードになります。レプリケーションの追加後に、最大 5 個のレプリカノードをクラスターに追加できます。

シャードがない Redis クラスターにレプリケーションを追加するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。

Redis エンジンを実行しているクラスターが一覧表示されます。

3. ノードを追加するクラスターの名前 (クラスター名の左にあるボックスではなく) を選択します。

レプリケーションが有効でない Redis クラスターは以下に該当します。

- クラスター化された Redis ではなく、Redis を実行しています。
- シャードがありません。

クラスターにシャードがある場合は、レプリケーションがすでに有効になっており、[ノードをクラスターに追加するには \(コンソール\)](#) を続行できます。

4. [Add replication] を選択します。
5. [レプリケーションの追加] で、このレプリケーションが有効なクラスターの説明を入力します。
6. [Add] (追加) を選択します。

クラスターのステータスが [available] になり次第、次の手順に進んでクラスターにレプリカを追加できます。

ノードをクラスターに追加するには (コンソール)

次の手順を使用して、クラスターにノードを追加できます。

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、ノードを追加する先のクラスターで実行されているエンジンを選択します。

選択したエンジンを実行しているクラスターが一覧表示されます。

3. クラスターのリストから、ノードを追加する先のクラスターの名前を選択します。

クラスターが Redis (クラスターモードが有効) クラスターの場合は、「[Redis \(クラスターモードが有効\) でのクラスターのスケーリング](#)」を参照してください。

クラスターがシャードがない Redis (クラスターモードが無効) クラスターの場合は、最初に「[シャードがない Redis クラスターにレプリケーションを追加するには](#)」のステップを完了してください。

4. [Add node] (ノードの追加) を選択します。
5. [ノードの追加] ダイアログボックスで、要求された情報を入力します。
6. このノードを直ちに追加する場合は [Apply Immediately (すぐに適用) - はい] ボタンを選択します。クラスターの次のメンテナンス期間にこのノードを追加する場合は [いいえ] を選択します。

保留中のリクエストに対する新しい追加および削除リクエストの影響

シナリオ	保留中のオペレーション	新しいリクエスト	結果
シナリオ 1	Delete	Delete	<p>新しい削除リクエスト (保留中または即時) は、保留中の削除リクエストを置き換えます。</p> <p>例えば、ノード 0001、0003、0007 が削除保留中で、ノード 0002 と 0004 を削除する新しいリクエストが発行された場合、ノード 0002 と 0004 のみが削除されます。ノード 0001、0003、0007 は削除されません。</p>
シナリオ 2	Delete	作成	<p>新しい作成リクエスト (保留中または直ちに) は、保留中の削除リクエストを置き換えます。</p> <p>例えば、ノード 0001、0003、0007 の削除が保留中で、ノードを作成する新しいリクエストが発行された場合、新しいノードが作成され、ノード 0001、0003、0007 は削除されません。</p>
シナリオ 3	作成	Delete	<p>新しい削除リクエスト (保留中または即時) は、保留中の作成リクエストを置き換えます。</p> <p>例えば、2 つのノードを作成する保留中の要求があり、ノード 0003 を削除する新しいリクエストが発行された場合、新しいノードは作成されず、ノード 0003 は削除されます。</p>

シナリオ	保留中のオペレーション	新しいリクエスト	結果
シナリオ 4	作成	作成	<p>新しい作成リクエストが保留中の作成リクエストに追加されます。</p> <p>例えば、2つのノードを作成する保留中のリクエストがあり、3つのノードを作成する新しいリクエストが発行された場合、新しいリクエストは保留中のリクエストに追加され、5つのノードが作成されます。</p> <div style="border: 1px solid #f08080; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>⚠ Important</p> <p>新しい作成リクエストが [直ちに適用 - はい] に設定されると、すべての作成リクエストが直ちに実行されます。新しい作成リクエストが [直ちに適用 - いいえ] に設定されると、すべての作成リクエストは保留中になります。</p> </div>

保留中のオペレーションを確認するには、[説明] タブを選択し、表示されている保留中の作成または削除の数を確認します。保留中の作成と保留中の削除の両方を持つことはできません。

7. [Add] ボタンを選択します。

しばらくすると、新しいノードが一覧表示され、ステータス [creating] になります。表示されない場合は、ブラウザのページを更新します。ノードのステータスが [available] (利用可能) に変わったら、新しいノードを使用できます。

AWS CLI を使用する場合

レプリケーションが有効でない既存の Redis (クラスターモードが無効) クラスターにノードを追加する場合は、最初にレプリケーショングループを作成して既存のクラスターをプライマリとして指定する必要があります。詳細については、「[使用可能な Redis キャッシュクラスターを使用した](#)

[レプリケーショングループの作成 \(AWS CLI\)](#)」を参照してください。レプリケーショングループが [available] になった後で、次のプロセスを続行できます。

AWS CLI を使用してクラスターにノードを追加するには、以下のパラメーターを指定して AWS CLI オペレーション `increase-replica-count` を使用します。

- `--replication-group-id` ノードを追加する先のレプリケーショングループの ID。
- `--new-replica-count` は、変更の適用後に、このレプリケーショングループに必要なとなるノードの数を指定します。このクラスターにノードを追加しても、`--new-replica-count` はこのクラスター内の現在ノードの数よりも大きくする必要があります。
- `--apply-immediately` または `--no-apply-immediately` は、次のメンテナンス時にこれらのノードを追加するかどうかを指定します。

Linux、macOS、Unix の場合:

```
aws elasticache increase-replica-count \  
  --replication-group-id my-replication-group \  
  --new-replica-count 4 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache increase-replica-count ^  
  --replication-group-id my-replication-group ^  
  --new-replica-count 4 ^  
  --apply-immediately
```

このオペレーションでは、次のような出力が生成されます (JSON 形式)。

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "node-test",  
    "Description": "node-test",  
    "Status": "modifying",  
    "PendingModifiedValues": {},  
    "MemberClusters": [  
      "node-test-001",  
      "node-test-002",  
      "node-test-003",  
      "node-test-004",
```



```
    "node-test-005"
  ],
  "NodeGroups": [
    {
      "NodeGroupId": "0001",
      "Status": "modifying",
      "PrimaryEndpoint": {
        "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "ReaderEndpoint": {
        "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "NodeGroupMembers": [
        {
          "CacheClusterId": "node-test-001",
          "CacheNodeId": "0001",
          "ReadEndpoint": {
            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
          },
          "PreferredAvailabilityZone": "us-west-2a",
          "CurrentRole": "primary"
        },
        {
          "CacheClusterId": "node-test-002",
          "CacheNodeId": "0001",
          "ReadEndpoint": {
            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
          },
          "PreferredAvailabilityZone": "us-west-2c",
          "CurrentRole": "replica"
        },
        {
          "CacheClusterId": "node-test-003",
          "CacheNodeId": "0001",
          "ReadEndpoint": {
            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
          }
        }
      ]
    }
  ]
}
```

```
        },
        "PreferredAvailabilityZone": "us-west-2b",
        "CurrentRole": "replica"
    }
]
}
],
"SnapshottingClusterId": "node-test-002",
"AutomaticFailover": "enabled",
"MultiAZ": "enabled",
"SnapshotRetentionLimit": 1,
"SnapshotWindow": "07:30-08:30",
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.large",
  "DataTiering": "disabled",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false,
"ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-test"
}
}
```

詳細については、AWS CLI のトピック「[increase-replica-count](#)」を参照してください。

ElastiCache API の使用

レプリケーションが有効でない既存の Redis (クラスターモードが無効) クラスターにノードを追加する場合は、最初にレプリケーショングループを作成して既存のクラスターをプライマリとして指定する必要があります。詳細については、「[スタンドアロン Redis \(クラスターモードが無効\) クラスターへのレプリカの追加 \(ElastiCache API\)](#)」を参照してください。レプリケーショングループが [available] になった後で、次のプロセスを続行できます。

ノードをクラスターに追加するには (ElastiCache API)

- 以下のパラメーターを指定して IncreaseReplicaCount API オペレーションを呼び出します。
 - ReplicationGroupId ノードを追加する先のクラスターの ID。
 - NewReplicaCount NewReplicaCount パラメータは、変更の適用後に、このクラスターに必要なノードの数を指定します。このクラスターにノードを追加しても、NewReplicaCount はこのクラスター内の現在ノードの数よりも大きくする必

必要があります。この値が現在のノードの数より少ない場合、そのノードの数がある DecreaseReplicaCount API を使用してクラスターから削除します。

- ApplyImmediately は、次のメンテナンス時にこれらのノードを追加するかどうかを指定します。
- Region ノードを追加するクラスターの AWS リージョンを指定します。

次の例は、クラスターにノードを追加する呼び出しを示しています。

Example

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=IncreaseReplicaCount  
  &ApplyImmediately=true  
  &NumCacheNodes=4  
  &ReplicationGroupId=my-replication-group  
  &Region=us-east-2  
  &Version=2014-12-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
  &X-Amz-Date=20141201T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

詳細については、「ElastiCache API トピック [IncreaseReplicaCount](#)」を参照してください。

クラスターからのノードの削除

AWS Management Console、AWS CLI、または ElastiCache API を使用して、クラスターからノードを削除できます。

AWS Management Console を使用する場合

クラスターからノードを削除するには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 右上にあるリストから、ノードを削除するクラスターの AWS リージョンを選択します。
3. ナビゲーションペインで、ノードを削除するクラスターで実行されているエンジンを選択します。

選択したエンジンを実行しているクラスターが一覧表示されます。

4. クラスターの一覧から、ノードを削除するクラスターの名前を選択します。

クラスターのノードが一覧表示されます。

5. 削除するノードのノード ID の左側にあるボックスをオンにします。ElastiCache コンソールで同時に削除できるノードは 1 つのみです。複数のノードを選択すると、[ノードの削除] ボタンは使用できません。

[ノードの削除] ページが表示されます。

6. ノードを削除するには、[ノードの削除] ページに入力し、[ノードの削除] を選択します。ノードを保持するには、[キャンセル] を選択します。

Important

ノードを削除すると、クラスターはマルチ AZ に準拠しなくなるため、[マルチ AZ] チェックボックスをオフにしてから、ノードを削除してください。[マルチ AZ] チェックボックスをオフにすると、[自動フェイルオーバー] を有効にすることができます。

保留中のリクエストに対する新しい追加および削除リクエストの影響

シナリオ	保留中のオペレーション	新しいリクエスト	結果
シナリオ 1	Delete	Delete	<p>新しい削除リクエスト (保留中または即時) は、保留中の削除リクエストを置き換えます。</p> <p>例えば、ノード 0001、0003、0007 が削除保留中で、ノード 0002 と 0004 を削除する新しいリクエストが発行された場合、ノード 0002 と 0004 のみが削除されます。ノード 0001、0003、0007 は削除されません。</p>
シナリオ 2	Delete	作成	<p>新しい作成リクエスト (保留中または直ちに) は、保留中の削除リクエストを置き換えます。</p> <p>例えば、ノード 0001、0003、0007 の削除が保留中で、ノードを作成する新しいリクエストが発行された場合、新しいノードが作成され、ノード 0001、0003、0007 は削除されません。</p>
シナリオ 3	作成	Delete	<p>新しい削除リクエスト (保留中または即時) は、保留中の作成リクエストを置き換えます。</p> <p>例えば、2 つのノードを作成する保留中の要求があり、ノード 0003 を削除する新しいリクエストが発行された場合、新しいノードは作成されず、ノード 0003 は削除されます。</p>
シナリオ 4	作成	作成	<p>新しい作成リクエストが保留中の作成リクエストに追加されます。</p> <p>例えば、2 つのノードを作成する保留中のリクエストがあり、3 つのノードを作成する新しいリクエストが発行された場合、新しいリクエストは保留中のリクエストに追加され、5 つのノードが作成されます。</p>

シナリオ	保留中のオペレーション	新しいリクエスト	結果
			<div style="border: 1px solid #f08080; padding: 10px;"> <p>⚠ Important</p> <p>新しい作成リクエストが [直ちに適用 - はい] に設定されると、すべての作成リクエストが直ちに実行されます。新しい作成リクエストが [直ちに適用 - いいえ] に設定されると、すべての作成リクエストは保留中になります。</p> </div>

保留中のオペレーションを確認するには、[説明] タブを選択し、表示されている保留中の作成または削除の数を確認します。保留中の作成と保留中の削除の両方を持つことはできません。

AWS CLI を使用する場合

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

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

- --replication-group-id ノードを削除するレプリケーショングループの ID。
- --new-replica-count --new-replica-count パラメータは、変更の適用後に、このクラスターに必要なとなるノードの数を指定します。
- --replicas-to-remove このクラスターから削除するノード ID のリスト。
- --apply-immediately または --no-apply-immediately は、次のメンテナンス時にこれらのノードを削除するかどうかを指定します。
- --region 複数のノードを削除するクラスターの AWS リージョンを指定します。

Note

このオペレーションを呼び出すときに、`--replicas-to-remove` または `--new-replica-count` パラメータの 1 つのみを渡すことができます。

Linux、macOS、Unix の場合:

```
aws elasticache decrease-replica-count \  
  --replication-group-id my-replication-group \  
  --new-replica-count 2 \  
  --region us-east-2 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache decrease-replica-count ^  
  --replication-group-id my-replication-group ^  
  --new-replica-count 3 ^  
  --region us-east-2 ^  
  --apply-immediately
```

このオペレーションでは、次のような出力が生成されます (JSON 形式)。

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "node-test",  
    "Description": "node-test"  
  },  
  "Status": "modifying",  
  "PendingModifiedValues": {},  
  "MemberClusters": [  
    "node-test-001",  
    "node-test-002",  
    "node-test-003",  
    "node-test-004",  
    "node-test-005",  
    "node-test-006"  
  ],  
}
```

```
"NodeGroups": [  
  {  
    "NodeGroupId": "0001",  
    "Status": "modifying",  
    "PrimaryEndpoint": {  
      "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",  
      "Port": 6379  
    },  
    "ReaderEndpoint": {  
      "Address": "node-test-  
ro.zzzzzz.ng.0001.usw2.cache.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeGroupMembers": [  
      {  
        "CacheClusterId": "node-test-001",  
        "CacheNodeId": "0001",  
        "ReadEndpoint": {  
          "Address": "node-  
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",  
          "Port": 6379  
        },  
        "PreferredAvailabilityZone": "us-west-2a",  
        "CurrentRole": "primary"  
      },  
      {  
        "CacheClusterId": "node-test-002",  
        "CacheNodeId": "0001",  
        "ReadEndpoint": {  
          "Address": "node-  
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",  
          "Port": 6379  
        },  
        "PreferredAvailabilityZone": "us-west-2c",  
        "CurrentRole": "replica"  
      },  
      {  
        "CacheClusterId": "node-test-003",  
        "CacheNodeId": "0001",  
        "ReadEndpoint": {  
          "Address": "node-  
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",  
          "Port": 6379  
        }  
      },  
    ]  
  },  
]
```



```
        "PreferredAvailabilityZone": "us-west-2b",
        "CurrentRole": "replica"
    },
    {
        "CacheClusterId": "node-test-004",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
            "Address": "node-
test-004.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
        },
        "PreferredAvailabilityZone": "us-west-2c",
        "CurrentRole": "replica"
    },
    {
        "CacheClusterId": "node-test-005",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
            "Address": "node-
test-005.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
        },
        "PreferredAvailabilityZone": "us-west-2b",
        "CurrentRole": "replica"
    },
    {
        "CacheClusterId": "node-test-006",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
            "Address": "node-
test-006.zzzzzz.0001.usw2.cache.amazonaws.com",
            "Port": 6379
        },
        "PreferredAvailabilityZone": "us-west-2b",
        "CurrentRole": "replica"
    }
]
}
},
"SnapshottingClusterId": "node-test-002",
"AutomaticFailover": "enabled",
"MultiAZ": "enabled",
"SnapshotRetentionLimit": 1,
"SnapshotWindow": "07:30-08:30",
```

```
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.large",
  "DataTiering": "disabled",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false,
"ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-
test"
  }
}
```

または、`decrease-replica-count` を呼び出すこともでき、また `--new-replica-count` パラメータを渡す代わりに、以下に示すように `--replicas-to-remove` パラメータを渡すことができます。

Linux、macOS、Unix の場合:

```
aws elasticache decrease-replica-count \
  --replication-group-id my-replication-group \
  --replicas-to-remove node-test-003 \
  --region us-east-2 \
  --apply-immediately
```

Windows の場合:

```
aws elasticache decrease-replica-count ^
  --replication-group-id my-replication-group ^
  --replicas-to-remove node-test-003 ^
  --region us-east-2 ^
  --apply-immediately
```

詳細については、「AWS CLI のトピック [decrease-replica-count](#)」を参照してください。

ElastiCache API の使用

ElastiCache API を使用してノードを削除するには、次のようにレプリケーショングループ ID と削除するノードの一覧を使用して `DecreaseReplicaCount` API オペレーションを呼び出します。

- `ReplicationGroupId` ノードを削除するレプリケーショングループの ID。
- `ReplicasToRemove` `ReplicasToRemove` パラメータは、変更の適用後に、このクラスターに必要なとなるノードの数を指定します。

- `ApplyImmediately` は、次のメンテナンス時にこれらのノードを削除するかどうかを指定します。
- `Region` 1 つのノードを削除するクラスターの AWS リージョンを指定します。

次の例では、`my-cluster` クラスターからノード `0004` と `0005` を直ちに削除します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=DecreaseReplicaCount  
  &ReplicationGroupId=my-replication-group  
  &ApplyImmediately=true  
  &ReplicasToRemove=node-test-003  
  &Region us-east-2  
  &Version=2014-12-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
  &X-Amz-Date=20141201T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

詳細については、「ElastiCache API トピック [DecreaseReplicaCount](#)」を参照してください。

保留中のノードの追加または削除オペレーションのキャンセル

変更を直ちに適用しないことを選択した場合、操作は、次のメンテナンス時間に実行されるまで pending ステータスになります。保留中のオペレーションはすべてキャンセルできます。

保留中のオペレーションをキャンセルするには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 右上のリストから、保留中のノードの追加または削除オペレーションをキャンセルする AWS リージョンを選択します。
3. ナビゲーションペインで、保留中のオペレーションをキャンセルするクラスターで実行されているエンジンを選択します。選択したエンジンを実行しているクラスターが一覧表示されます。
4. クラスターのリストで、キャンセルする保留中のオペレーションがあるクラスターの名前 (クラスター名の左にあるボックスではなく) を選択します。
5. 保留中のオペレーションを確認するには、[説明] タブを選択し、表示されている保留中の作成または削除の数を確認します。保留中の作成と保留中の削除の両方を持つことはできません。
6. [Nodes] タブを選択します。
7. すべての保留中のオペレーションをキャンセルするには、[Cancel Pending] をクリックします。[Cancel Pending] ダイアログボックスが表示されます。
8. [Cancel Pending] ボタンを選択して、すべての保留中のオペレーションをキャンセルすることを確認します。オペレーションを保持する場合は、[Cancel] を選択します。

クラスターの削除

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

Warning

ElastiCache for Redis クラスターを削除しても、手動スナップショットは保持されます。クラスターを削除する前に最終スナップショットを作成することもできます。自動キャッシュスナップショットは保持されません。

AWS Management Console を使用する場合は

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

クラスターを削除するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ElastiCache コンソールのダッシュボードで、削除するクラスターで実行されているエンジンを選択します。

そのエンジンを実行しているすべてのクラスターが一覧表示されます。

3. 削除するクラスターを選択するには、クラスターのリストからそのクラスターの名前を選択します。

Important

ElastiCache コンソールから、一度に 1 つずつクラスターを削除できます。複数のクラスターを選択すると、削除オペレーションが無効になります。

4. [Actions] (アクション) として、[Delete] (削除) を選択します。
5. [クラスターの削除] 確認画面で、[削除] を選択してクラスターを削除するか、[キャンセル] を選択してクラスターを保持します。

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

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

AWS CLI を使用する場合

次のコードでは、キャッシュクラスター `my-cluster` を削除します。

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

`delete-cache-cluster` CLI アクションは 1 つのキャッシュクラスターのみを削除します。複数のキャッシュクラスターを削除する場合は、削除するキャッシュクラスターごとに `delete-cache-cluster` を呼び出します。1 つのキャッシュクラスターの削除が終了するまで待たなくても次のクラスターを削除できます。

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-cluster \  
  --cache-cluster-id my-cluster \  
  --region us-east-2
```

Windows の場合:

```
aws elasticache delete-cache-cluster ^  
  --cache-cluster-id my-cluster ^  
  --region us-east-2
```

詳細については、「AWS CLI for ElastiCache トピック [delete-cache-cluster](#)」を参照してください。

ElastiCache API の使用

次のコードはクラスター `my-cluster` を削除します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DeleteCacheCluster  
&CacheClusterId=my-cluster  
&Region us-east-2  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T220302Z  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Date=20150202T220302Z
```

```
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20150202T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

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

詳細については、「ElastiCache API リファレンストピック [DeleteCacheCluster](#)」を参照してください。

クラスターまたはレプリケーショングループへのアクセス

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

Amazon Virtual Private Cloud (Amazon VPC) 内で ElastiCache インスタンスを起動した場合は、同じ Amazon VPC 内の Amazon EC2 インスタンスから ElastiCache インスタンスにアクセスできます。または、VPC ピアリングを使用して、異なる Amazon VPC 内の Amazon EC2 から ElastiCache インスタンスにアクセスできます。

ElastiCache インスタンスを EC2 Classic で起動した場合、そのインスタンスに関連付けられた Amazon EC2 セキュリティグループに、キャッシュセキュリティグループへのアクセスを許可することで、EC2 インスタンスにクラスターへのアクセスを許可することができます。デフォルトでは、クラスターへのアクセスはそのクラスターを起動したアカウントに制限されています。

トピック

- [クラスターまたはレプリケーショングループへのアクセスの許可](#)

クラスターまたはレプリケーショングループへのアクセスの許可

EC2-VPC でクラスターを起動した

Amazon Virtual Private Cloud (Amazon VPC) でクラスターを起動した場合、同じ Amazon VPC で実行されている Amazon EC2 インスタンスからのみ ElastiCache クラスターに接続できます。この場合、クラスターに対するネットワーク進入を許可する必要があります。


Note

[Local Zones] を使用している場合、それを有効にしていることを確認します。詳細については、「[Local Zones の有効化](#)」を参照してください。これにより、VPC はそのローカルゾーンに拡張され、VPC はそのサブネットを他のアベイラビリティーゾーン、関連するゲートウェイ、ルートテーブル、およびその他のセキュリティグループの考慮事項のサブネットとして扱い、自動的に調整されます。

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

1. AWS Management Console にサインインし、Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2/>) を開きます。

2. ナビゲーションペインで、[Network & Security] の下にある [Security Groups] を選択します。
3. セキュリティグループのリストから、Amazon VPC のセキュリティグループを選択します。ElastiCache 用のセキュリティグループを作成した場合を除き、このセキュリティグループは、[default] という名前になります。
4. Inbound タブを選択し、次の操作を行います。
 - a. Edit (編集) を選択します。
 - b. ルールの追加 を選択します。
 - c. Type 列で Custom TCP rule を選択します。
 - d. Port range ボックスに、クラスターノードのポート番号を入力します。この番号は、クラスターの起動時に指定した番号と同じ番号である必要があります。Redis のデフォルトポートは **6379** です。
 - e. [Source] ボックスで [Anywhere] を選択します。ポート範囲が 0.0.0.0/0 になるため、Amazon VPC 内で起動したすべての Amazon EC2 インスタンスを ElastiCache ノードに接続できます。

 Important

ElastiCache クラスターを 0.0.0.0/0 に設定すると、パブリック IP アドレスが割り当てられず、VPC 外からアクセスできないため、クラスターはインターネットに公開されません。ただし、お客様のアカウントの他の Amazon EC2 インスタンスにデフォルトのセキュリティグループが適用され、そのインスタンスにパブリック IP アドレスが付与される場合があります。それがデフォルトポートで何かを実行している場合、そのサービスが意図せず公開されることがあります。そのため、ElastiCache でのみ使用される VPC のセキュリティグループを作成されることをお勧めします。詳細については、「[カスタムセキュリティグループ](#)」を参照してください。

- f. 保存 を選択します。

Amazon VPC で Amazon EC2 インスタンスを起動すると、そのインスタンスは ElastiCache クラスターに接続できます。

AWS 外部からの ElastiCache リソースへのアクセス

Amazon ElastiCache は、クラウドベースのメモリ内のキー値ストアを提供する AWS サービスです。このサービスは、AWS 内からのみアクセスできるように設計されています。ElastiCache クラスターが VPC 内でホストされている場合は、ネットワークアドレス変換 (NAT) インスタンスを使用して外部からアクセスできるようにできます。

要件

AWS 外部から ElastiCache リソースにアクセスするには、以下の要件を満たしている必要があります。

- クラスターが VPC 内にあり、ネットワークアドレス変換 (NAT) インスタンスを介してアクセスできる必要があります。これは必須の要件であり、例外はありません。
- NAT インスタンスがクラスターと同じ VPC 内で起動されている必要があります。
- NAT インスタンスがクラスターとは別のパブリックサブネットに起動されている必要があります。
- Elastic IP (EIP) アドレスが NAT インスタンスに関連付けられている必要があります。NAT インスタンスのポートは、iptables のポート転送機能を使用して VPC 内のキャッシュノードポートに転送されます。

考慮事項

ElastiCache 外部から ElastiCache リソースにアクセスする際には、以下の点を考慮してください。

- クライアントは EIP に接続し、NAT インスタンスのポートをキャッシュします。NAT インスタンスでポート転送すると、トラフィックは適切なキャッシュクラスターノードに転送されます。
- クラスターノードが追加または交換されたら、この変更が反映されるように iptables ルールが更新される必要があります。

制限事項

このアプローチは、テストおよび開発の目的にしか使用できません。以下の制限があるため、本稼働で使用することは推奨されません。

- NAT インスタンスは、クライアントと複数のクラスター間のプロキシとして機能します。プロキシを追加すると、キャッシュクラスターのパフォーマンスに影響が及びます。この影響は、NAT インスタンスを介してアクセスするキャッシュクラスターの数に応じて増大します。

- クライアントから NAT インスタンスへのトラフィックは暗号化されません。したがって、NAT インスタンスを介して機密データを送信することは回避してください。
- NAT インスタンスは、別のインスタンスを維持するためのオーバーヘッドを増加させます。
- NAT インスタンスは単一障害点として機能します。VPC で高可用性 NAT をセットアップする方法については、「[Amazon VPC NAT インスタンスの高可用性: 例](#)」を参照してください。

AWS 外部からの ElastiCache リソースにアクセスする方法

次の手順は、NAT インスタンスを使用して ElastiCache リソースに接続する方法を示しています。

これらのステップは、以下を前提としています。

- ```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to 10.0.1.231:6379
```
- ```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to 10.0.1.232:6379
```

次に、逆方向の NAT が必要です。

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.0.55
```

デフォルトで無効になっている IP 転送も有効にする必要があります。

```
sudo sed -i 's/net.ipv4.ip_forward=0/net.ipv4.ip_forward=1/g' /etc/sysctl.conf sudo sysctl --system
```

- 以下を使って、Redis クラスターにアクセスします。
 - IP アドレス – 10.0.1.230
 - デフォルトの Redis ポート – 6379
 - セキュリティグループ – sg-bd56b7da
 - AWS インスタンス IP アドレス — sg-bd56b7da
- 信頼済みクライアントの IP アドレスが 198.51.100.27 である。
- NAT インスタンスの Elastic IP アドレスが 203.0.113.73 である。
- NAT インスタンスのセキュリティグループが sg-ce56b7a9 である。

NAT インスタンスを使用して ElastiCache リソースに接続するには

1. キャッシュクラスターと同じ VPC 内のパブリックサブネット内に NAT インスタンスを作成します。

デフォルトでは、VPC ウィザードが `cache.m1.small` ノードタイプを起動します。必要に応じてノードサイズを選択する必要があります。AWS 外部から ElastiCache にアクセスできるようにするには、EC2 NAT AMI を使用する必要があります。

NAT インスタンスの作成については、「AWS VPC ユーザーガイド」の「[NAT インスタンス](#)」を参照してください。

2. キャッシュクラスターと NAT インスタンスのセキュリティグループルールを作成します。

NAT インスタンスのセキュリティグループとクラスターインスタンスには以下のルールが必要です。

- 2つのインバウンドルール

- 信頼済みクライアントから、NAT インスタンスから転送された各キャッシュポート (6379 - 6381) への TCP 接続を許可するルール。
- 信頼済みクライアントへの SSH アクセスを許可するルール。

NAT インスタンスのセキュリティグループ - インバウンドルール

タイプ	プロトコル	ポート範囲	ソース
カスタム TCP ルール	TCP	6379-6380	198.51.100.27/32
SSH	TCP	22	203.0.113.73/32

- キャッシュポート (6379) への TCP 接続を許可するアウトバウンドルール。

NAT インスタンスのセキュリティグループ - アウトバウンドルール

タイプ	プロトコル	ポート範囲	デスティネーション
カスタム TCP ルール	TCP	6379	sg-ce56b7a9 (クラスターインスタンスのセキュリティグループ)

- NAT インスタンスからキャッシュポート (6379) への TCP 接続を許可するクラスターのセキュリティグループのインバウンドルール。

クラスターインスタンスのセキュリティグループ - インバウンドルール

タイプ	プロトコル	ポート範囲	ソース
カスタム TCP ルール	TCP	6379	sg-bd56b7da (クラスターセキュリティグループ)

3. ルールを検証します。

- 信頼済みクライアントが NAT インスタンスに SSH 接続できることを確認します。
- 信頼済みクライアントが NAT インスタンスからクラスターに接続できることを確認します。

4. NAT インスタンスに iptables ルールを追加します。

NAT インスタンスからのキャッシュポートをクラスターノードに転送するには、iptables ルールをクラスターのノードごとに NAT テーブルに追加する必要があります。以下に例を示します。

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to 10.0.1.230:6379
```

ポート番号は、クラスターのノードごとに一意である必要があります。たとえば、ポート 11211 ~ 11213 を使用する 3 つのノードで構成される Redis クラスターを使用している場合、ルールは次のようになります。

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to 10.0.1.230:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to 10.0.1.231:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to 10.0.1.232:6379
```

5. 信頼済みクライアントがクラスターに接続できることを確認します。

信頼済みクライアントは、NAT インスタンスに関連付けられている EIP と、適切なクラスターノードに対応するクラスターポートに接続する必要があります。たとえば、PHP の接続文字列は次のようになります。

```
redis->connect( '203.0.113.73', 6379 );
redis->connect( '203.0.113.73', 6380 );
redis->connect( '203.0.113.73', 6381 );
```

telnet クライアントを使用して接続を検証することもできます。例:

```
telnet 203.0.113.73 6379
telnet 203.0.113.73 6380
telnet 203.0.113.73 6381
```

6. iptables 設定を保存します。

ルールをテストし、検証してから保存します。Red Hat ベースの Linux ディストリビューション (Amazon Linux など) を使用している場合は、次のコマンドを実行します。

```
service iptables save
```

関連トピック

以下のトピックも役に立つ場合があります。

- [Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)
- [お客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)
- [NAT インスタンス](#)
- [ElastiCache クライアントの設定](#)
- [Amazon VPC NAT インスタンスの高可用性: 例](#)

接続エンドポイントの検索

エンドポイントを使用してアプリケーションがクラスターに接続します。エンドポイントはノードまたはクラスターの一意的なアドレスです。

自動検出を使用しない場合は、読み取りと書き込みに個々のノードのエンドポイントを使用するようにクライアントを設定する必要があります。また、ノードの追加や削除時にはそれらのエンドポイントを更新する必要があります。

使用するエンドポイント

- Redis スタンドアロンノード。ノードのエンドポイントを読み取りと書き込みの両方のオペレーションに使用します。
- Redis (クラスターモードが無効) クラスター、プライマリエンドポイントをすべての書き込みオペレーションに使用します。読み込みエンドポイントを使用して、すべてのリードレプリカ間でエンドポイントへの着信接続を均等に分割します。個々のノードエンドポイント (API/CLI ではリードエンドポイント) を読み取りオペレーションに使用します。
- Redis (クラスターモードが有効) クラスター、クラスターの設定エンドポイントを、クラスターモードが有効コマンドをサポートするすべてのオペレーションに使用します。Redis クラスター (Redis 3.2) をサポートするクライアントを使用する必要があります。個々のノードエンドポイント (API/CLI ではリードエンドポイント) から読み取ることもできます。

以下のセクションで、実行するエンジンに必要なエンドポイントの検索について説明します。

Redis (クラスターモードが無効) クラスターのエンドポイント (コンソール)

Redis (クラスターモードが無効) クラスターに 1 つのみのノードがある場合、ノードのエンドポイントは読み取りと書き込みの両方に使用されます。Redis (クラスターモードが無効) クラスターに複数のノードがある場合は、[プライマリエンドポイント]、[読み込みエンドポイント]、[ノードエンドポイント] の 3 種類のエンドポイントがあります。

プライマリエンドポイントは、常にクラスターのプライマリノードに解決される DNS 名です。プライマリエンドポイントは、リードレプリカのプライマリロールへの昇格など、クラスターに対する変更の影響を受けません。書き込みアクティビティの場合、アプリケーションをプライマリエンドポイントに接続することをお勧めします。

読み込みエンドポイントによって、ElastiCache for Redis クラスター内のすべてのリードレプリカ間でエンドポイントへの着信接続が均等に分割されます。アプリケーションがいつ接続を作成するか、アプリケーションが接続をどのように (再) 利用するかなどの追加要因によって、トラフィックの分散が決定されます。レプリカが追加または削除されても、読み込みエンドポイントはリアルタイムでクラスターの変更に対応します。ElastiCache for Redis クラスターの複数のリードレプリカを異なる AWS アベイラビリティゾーン (AZ) に配置して、リーダーエンドポイントの高可用性を確保することができます。

Note

リーダーエンドポイントはロードバランサーではありません。これは、ラウンドロビン方式でレプリカノードの 1 つの IP アドレスに解決される DNS レコードです。

読み取りアクティビティの場合、アプリケーションはクラスター内のいずれのノードにも接続できます。プライマリエンドポイントとは異なり、ノードエンドポイントは特定のエンドポイントに解決されます。レプリカの追加または削除など、クラスターに変更を加えた場合は、アプリケーションでノードエンドポイントを更新する必要があります。

Redis (クラスターモードが無効) クラスターのエンドポイントを検索するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。

クラスター画面には Redis (クラスターモード無効) クラスターと Redis (クラスターモード有効) クラスターのリストが表示されます。

3. クラスターのプライマリエンドポイントや読み込みエンドポイントを検索するには、クラスターの名前 (左にあるボタンではない) を選択します。

▼ Cluster details

Cluster name [Redacted]	Description [Redacted]	Node type cache.r6g.large	Status Available
Engine Redis	Engine version 6.0.5	Global datastore -	Global datastore role -
Update status Update available	Cluster mode Off	Shards 1	Number of nodes 3
Data tiering Disabled	Multi-AZ Enabled	Auto-failover Enabled	Encryption in transit Disabled
Encryption at rest Disabled	Parameter group default.redis6.x	Outpost ARN -	Configuration endpoint -
Primary endpoint [Redacted] -encrypted.llru6f.ng.0001.use1.cache.amazonaws.com:6379	Reader endpoint [Redacted] -encrypted-ro.llru6f.ng.0001.use1.cache.amazonaws.com:6379	ARN [Redacted]	

Redis (クラスターモードが無効) クラスターのプライマリエンドポイント

クラスターに 1 つのみのノードがある場合、プライマリエンドポイントはないため、次のステップに進むことができます。

4. Redis (クラスターモードが無効) クラスターにレプリカノードがある場合は、クラスターの名前を選択してから [Nodes] (ノード) タブを選択して、クラスターのレプリカノードエンドポイントを検索できます。

ノードの画面では、クラスター内のプライマリとレプリカの各ノードがそのエンドポイントと共に表示されます。

<input type="checkbox"/>	Node Name	Status	Current Role	Port	Endpoint
<input type="checkbox"/>	test-no-001	available	primary	6379	[Redacted] amazonaws.com
<input type="checkbox"/>	test-no-002	available	replica	6379	[Redacted] amazonaws.com
<input type="checkbox"/>	test-no-003	available	replica	6379	[Redacted] amazonaws.com

Redis (クラスターモードが無効) クラスターのノードエンドポイント

5. エンドポイントをクリップボードにコピーするには:
- 一度に 1 つのみ、コピーするエンドポイントを見つけます。
 - エンドポイントアドレスのすぐ前にあるコピーアイコンを選択します。

エンドポイントがクリップボードにコピーされます。エンドポイントを使用してノードに接続する方法については、「[ノードに接続する](#)」を参照してください。

Redis (クラスターモードが無効) プライマリエンドポイントは以下のように表示されます。転送時の暗号化が有効かどうかによって違いがあります。

転送時の暗号化が無効

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port
```

```
redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

転送時の暗号化が有効

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port
```

```
master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

Redis (クラスターモードが有効) クラスターのエンドポイントを検索する (コンソール)

Redis (クラスターモードが有効) クラスターには、単一の設定エンドポイントがあります。設定エンドポイントに接続することで、アプリケーションはクラスター内のシャードごとにプライマリおよびリードエンドポイントを検出できます。

Redis (クラスターモードが有効) クラスターのエンドポイントを検索するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。

クラスター画面には Redis (クラスターモード無効) クラスターと Redis (クラスターモード有効) クラスターのリストが表示されます。接続する Redis (クラスターモードが有効) クラスターを選択します。

3. クラスターの設定エンドポイントを検索するには、ラジオボタンではなくクラスターの名前を選択します。

4. [Configuration endpoint] (設定エンドポイント) は [Cluster details] (クラスターの詳細) の下に表示されます。コピーするには、エンドポイントの左側にある [copy] (コピー) アイコンを選択します。

エンドポイントの検索 (AWS CLI)

Amazon ElastiCache の AWS CLI を使用して、ノード、クラスター、レプリケーショングループのエンドポイントを検索できます。

トピック

- [ノードとクラスターのエンドポイントの検索\(AWS CLI\)](#)
- [レプリケーショングループのエンドポイントの検索 \(AWS CLI\)](#)

ノードとクラスターのエンドポイントの検索(AWS CLI)

AWS CLI で `describe-cache-clusters` コマンドを使用して、クラスターとそのノードのエンドポイントを検出できます。Redis クラスターでは、そのコマンドがクラスターエンドポイントを返します。オプションのパラメータ `--show-cache-node-info` を含めた場合、コマンドはクラスター内の個々のノードにエンドポイントを返します。

Example

次のコマンドは、単一ノードの Redis (クラスターモードが無効) クラスター `mycluster` に関するクラスター情報を取得します。

Important

`--cache-cluster-id` パラメータは、単一ノードの Redis (クラスターモードが無効) クラスター ID または Redis レプリケーショングループ内の特定のノード ID で使用できます。Redis レプリケーショングループの `--cache-cluster-id` は、`0001` のような 4 桁の値です。`--cache-cluster-id` が Redis レプリケーショングループ内のクラスター (ノード) の ID である場合は、`replication-group-id` が出力に含まれます。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-clusters \  
  --cache-cluster-id redis-cluster \  
  --show-cache-node-info
```

Windows の場合:

```
aws elasticache describe-cache-clusters ^
```

```
--cache-cluster-id redis-cluster ^  
--show-cache-node-info
```

上記のオペレーションからの出力は以下のような JSON 形式になります。

```
{  
  "CacheClusters": [  
    {  
      "CacheClusterStatus": "available",  
      "SecurityGroups": [  
        {  
          "SecurityGroupId": "sg-77186e0d",  
          "Status": "active"  
        }  
      ],  
      "CacheNodes": [  
        {  
          "CustomerAvailabilityZone": "us-east-1b",  
          "CacheNodeCreateTime": "2018-04-25T18:19:28.241Z",  
          "CacheNodeStatus": "available",  
          "CacheNodeId": "0001",  
          "Endpoint": {  
            "Address": "redis-cluster.amazonaws.com",  
            "Port": 6379  
          },  
          "ParameterGroupStatus": "in-sync"  
        }  
      ],  
      "AtRestEncryptionEnabled": false,  
      "CacheClusterId": "redis-cluster",  
      "TransitEncryptionEnabled": false,  
      "CacheParameterGroup": {  
        "ParameterApplyStatus": "in-sync",  
        "CacheNodeIdsToReboot": [],  
        "CacheParameterGroupName": "default.redis3.2"  
      },  
      "NumCacheNodes": 1,  
      "PreferredAvailabilityZone": "us-east-1b",  
      "AutoMinorVersionUpgrade": true,  
      "Engine": "redis",  
      "AuthTokenEnabled": false,  
      "PendingModifiedValues": {},  
      "PreferredMaintenanceWindow": "tue:08:30-tue:09:30",  
    }  
  ]  
}
```

```
    "CacheSecurityGroups": [],
    "CacheSubnetGroupName": "default",
    "CacheNodeType": "cache.t2.small",
    "DataTiering": "disabled"
    "EngineVersion": "3.2.10",
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
    "CacheClusterCreateTime": "2018-04-25T18:19:28.241Z"
  }
]
}
```

詳細については、トピック「[describe-cache-clusters](#)」を参照してください。

レプリケーショングループのエンドポイントの検索 (AWS CLI)

AWS CLI で `describe-replication-groups` コマンドを使用して、レプリケーショングループとそのクラスターのエンドポイントを検出できます。このコマンドでは、読み込みエンドポイントと合わせて、レプリケーショングループのプライマリエンドポイント、レプリケーショングループ内のすべてのクラスター (ノード)、およびそのエンドポイントのリストが返ります。

次のオペレーションでは、レプリケーショングループ `myreplgroup` のプライマリエンドポイントと読み込みエンドポイントが取得されます。すべての書き込みオペレーションにプライマリエンドポイントを使用します。

```
aws elasticache describe-replication-groups \
  --replication-group-id myreplgroup
```

Windows の場合:

```
aws elasticache describe-replication-groups ^
  --replication-group-id myreplgroup
```

このオペレーションからの出力は以下のような JSON 形式になります。

```
{
  "ReplicationGroups": [
    {
      "Status": "available",
      "Description": "test",
      "NodeGroups": [
        {
```

```
    "Status": "available",
    "NodeGroupMembers": [
      {
        "CurrentRole": "primary",
        "PreferredAvailabilityZone": "us-west-2a",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address": "myreplgroup-001.amazonaws.com"
        },
        "CacheClusterId": "myreplgroup-001"
      },
      {
        "CurrentRole": "replica",
        "PreferredAvailabilityZone": "us-west-2b",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address": "myreplgroup-002.amazonaws.com"
        },
        "CacheClusterId": "myreplgroup-002"
      },
      {
        "CurrentRole": "replica",
        "PreferredAvailabilityZone": "us-west-2c",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address": "myreplgroup-003.amazonaws.com"
        },
        "CacheClusterId": "myreplgroup-003"
      }
    ],
    "NodeGroupId": "0001",
    "PrimaryEndpoint": {
      "Port": 6379,
      "Address": "myreplgroup.amazonaws.com"
    },
    "ReaderEndpoint": {
      "Port": 6379,
      "Address": "myreplgroup-ro.amazonaws.com"
    }
  }
],
```

```
    "ReplicationGroupId": "myreplgroup",
    "AutomaticFailover": "enabled",
    "SnapshottingClusterId": "myreplgroup-002",
    "MemberClusters": [
      "myreplgroup-001",
      "myreplgroup-002",
      "myreplgroup-003"
    ],
    "PendingModifiedValues": {}
  }
]
}
```

詳細については、AWS CLI コマンドリファレンスの [describe-replication-groups](#) を参照してください。

エンドポイントの検索 (ElastiCache API)

Amazon ElastiCache API を使用して、ノード、クラスター、レプリケーショングループのエンドポイントを検索できます。

トピック

- [ノードとクラスターのエンドポイントの検索 \(ElastiCache API\)](#)
- [レプリケーショングループのエンドポイントの検索 \(ElastiCache API\)](#)

ノードとクラスターのエンドポイントの検索 (ElastiCache API)

ElastiCache API を使用して、DescribeCacheClusters アクションでクラスターのエンドポイントとそのノードを検出することができます。Redis クラスターでは、そのコマンドがクラスターエンドポイントを返します。オプションのパラメータ ShowCacheNodeInfo を含めた場合、アクションはクラスター内の個々のノードのエンドポイントも返します。

Example

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterId=mycluster  
&ShowCacheNodeInfo=true  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

レプリケーショングループのエンドポイントの検索 (ElastiCache API)

ElastiCache API を使用して、DescribeReplicationGroups アクションでレプリケーショングループのエンドポイントとそのクラスターを検出できます。このアクションでは、読み込みエンドポイントに合わせて、レプリケーショングループのプライマリエンドポイント、レプリケーショングループのすべてのクラスター、およびそのエンドポイントのリストが返ります。

以下のオペレーションでは、レプリケーショングループ myreplgroup のプライマリエンドポイント (PrimaryEndpoint)、読み込みエンドポイント (ReaderEndpoint)、個々のノードのエンドポイント (ReadEndpoint) を取得します。すべての書き込みオペレーションにプライマリエンドポイントを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeReplicationGroups  
&ReplicationGroupId=myreplgroup  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

詳細については、「[DescribeReplicationGroups](#)」を参照してください。

シャードの使用

シャード (API/CLI: ノードグループ) は、1~6 個の Redis ノードで構成される集合です。Redis (クラスターモードが無効) クラスターに、複数のシャード (シャード) があることはありません。シャードを使用すると、大規模なデータベースを、データシャードと呼ばれるより小さく、高速で、より簡単に管理できるパーツに分割できます。これにより、オペレーションを複数の個別のセクションに分散することで、データベース効率が向上します。シャードを使用すると、パフォーマンス、スケーラビリティ、コスト効率の向上など、多くのメリットが得られます。

シャードの数が多くレプリカの数が多いクラスターを作成できます。クラスターあたり最大 90 ノードです。このクラスター設定は、シャード 90 個およびレプリカ 0 個からシャード 15 個およびレプリカ 5 個 (許容されるレプリカの最大数) までです。クラスターのデータは、クラスターのシャード間で分割されます。シャードに複数のノードがある場合、1 つを読み書きのプライマリノード、その他を読み取り専用のレプリカノードとするレプリケーションが実装されます。

Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

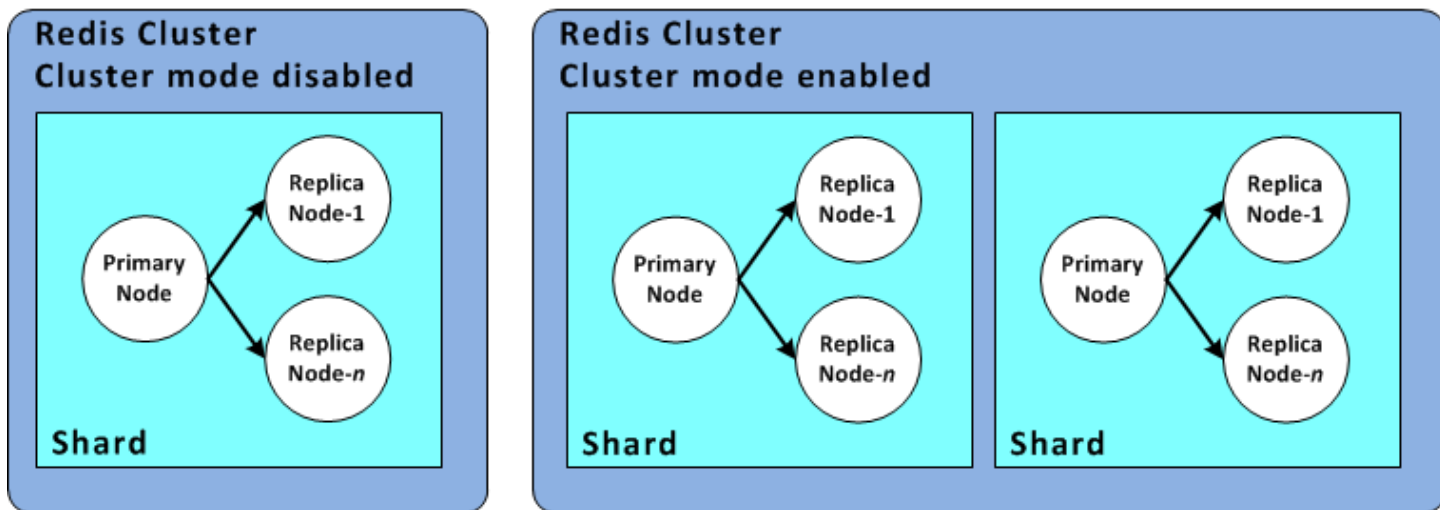
5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

ElastiCache コンソールを使用して Redis (クラスターモードが有効) クラスターを作成するときは、クラスター内のシャード数とシャード内のノード数を指定します。詳細については、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。ElastiCache API または を使用してクラスター AWS CLI を作成する場合 (API/CLI ではレプリケーショングループと呼ばれます)、シャード内のノード数 (API/CLI: ノードグループ) を個別に設定できます。詳細については、次を参照してください。

- API: [CreateReplicationGroup](#)
- CLI: [create-replication-group](#)

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



Redis シャードの構成

詳細については、[Redis \(クラスターモードが有効\) のオフラインの再分散とシャードの再分散およびRedis 用のオンラインリシャーディングおよびシャードの再分散 \(クラスターモードが有効\)](#)を参照してください。

シャードの ID を見つける

シャードの ID は、AWS Management Console、AWS CLI または ElastiCache API を使用して検索できます。

の使用 AWS Management Console

トピック

- [Redis \(クラスターモードが無効\) の場合](#)
- [Redis \(クラスターモードが有効\) の場合](#)

Redis (クラスターモードが無効) の場合

Redis (クラスターモードが無効) レプリケーショングループシャード ID は、常に 0001 です。

Redis (クラスターモードが有効) の場合

次の手順では AWS Management Console 、 を使用して Redis (クラスターモードが有効) のレプリケーショングループのシャード ID を検索します。

Redis (クラスターモードが有効) レプリケーショングループ内のシャード ID を見つけるには

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis] を選択し、シャード ID を見つけたい Redis (クラスターモードが有効) レプリケーショングループの名前を選択します。
3. [Shard Name (シャード名)] 列で、シャード ID はシャード名の末尾 4 桁の数字です。

の使用 AWS CLI

Redis (クラスターモードが無効) または Redis (クラスターモードが有効) レプリケーショングループのシャード (ノードグループ) ID を検索するには、次のオプションパラメータ `describe-replication-groups` を指定して AWS CLI オペレーションを使用します。

- **`--replication-group-id`**—指定されたレプリケーショングループの詳細への出力を制限するときに使用するオプションのパラメータ。このパラメータを省略すると、最大 100 個のレプリケーショングループの詳細が返されます。

Example

このコマンドは、`sample-repl-group` の詳細を返します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-replication-groups \
```

```
--replication-group-id sample-repl-group
```

Windows の場合:

```
aws elasticache describe-replication-groups ^  
--replication-group-id sample-repl-group
```

このコマンドによる出力は次のようになります。シャード (ノードグループ) ID は、見つけやすいように#####されます。

```
{  
  "ReplicationGroups": [  
    {  
      "Status": "available",  
      "Description": "2 shards, 2 nodes (1 + 1 replica)",  
      "NodeGroups": [  
        {  
          "Status": "available",  
          "Slots": "0-8191",  
          "NodeGroupId": "0001",  
          "NodeGroupMembers": [  
            {  
              "PreferredAvailabilityZone": "us-west-2c",  
              "CacheNodeId": "0001",  
              "CacheClusterId": "sample-repl-group-0001-001"  
            },  
            {  
              "PreferredAvailabilityZone": "us-west-2a",  
              "CacheNodeId": "0001",  
              "CacheClusterId": "sample-repl-group-0001-002"  
            }  
          ]  
        },  
        {  
          "Status": "available",  
          "Slots": "8192-16383",  
          "NodeGroupId": "0002",  
          "NodeGroupMembers": [  
            {  
              "PreferredAvailabilityZone": "us-west-2b",  
              "CacheNodeId": "0001",  
              "CacheClusterId": "sample-repl-group-0002-001"  
            }  
          ],  
        }  
      ]  
    }  
  ]  
}
```

```
        {
            "PreferredAvailabilityZone": "us-west-2a",
            "CacheNodeId": "0001",
            "CacheClusterId": "sample-repl-group-0002-002"
        }
    ]
}
],
"ConfigurationEndpoint": {
    "Port": 6379,
    "Address": "sample-repl-
group.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
},
"ClusterEnabled": true,
"ReplicationGroupId": "sample-repl-group",
"SnapshotRetentionLimit": 1,
"AutomaticFailover": "enabled",
"SnapshotWindow": "13:00-14:00",
"MemberClusters": [
    "sample-repl-group-0001-001",
    "sample-repl-group-0001-002",
    "sample-repl-group-0002-001",
    "sample-repl-group-0002-002"
],
"CacheNodeType": "cache.m3.medium",
"DataTiering": "disabled",
"PendingModifiedValues": {}
}
]
}
```

ElastiCache API の使用

Redis (クラスターモードが無効) または Redis (クラスターモードが有効) レプリケーショングループのシャード (ノードグループ) ID を検索するには、次のオプションパラメータ `describe-replication-groups` を指定して AWS CLI オペレーションを使用します。

- **ReplicationGroupId**—指定されたレプリケーショングループの詳細への出力を制限するときに使用するオプションのパラメータ。このパラメータを省略すると、最大 **xxx** 個のレプリケーショングループの詳細が返されます。

Example

このコマンドは、sample-repl-group の詳細を返します。

Linux、macOS、Unix の場合:

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeReplicationGroup  
&ReplicationGroupId=sample-repl-group  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

Memcached キャッシュと Redis の独自設計型キャッシュの比較

Amazon は Memcached および Redis キャッシュエンジン ElastiCache をサポートしています。各エンジンにはいくつかのメリットがあります。このトピックの情報を参考にして、要件を満たす最適なエンジンとバージョンを選択してください。

Important

キャッシュ、独自設計型クラスター、またはレプリケーショングループを作成したら、新しいエンジンバージョンにアップグレードできますが、古いエンジンバージョンにダウングレードすることはできません。古いエンジンバージョンを使用する場合は、既存のキャッシュ、独自設計型クラスター、またはレプリケーショングループを削除し、以前のエンジンバージョンで再度作成する必要があります。

見かけ上エンジンは似ています。それぞれのエンジンは、インメモリキー/値ストアです。ただし、実際には大きな違いがあります。

以下がお客様の状況に当てはまる場合は、Memcached を選択します。

- できるだけシンプルなモデルが必要である。
- 複数のコアまたはスレッドを持つ大きなノードを実行する必要がある。
- システムでの需要の増減に応じてノードを追加または削除するスケールアウトおよびスケールイン機能が必要である。

- オブジェクトをキャッシュする必要があります。

次の条件が当てはまる場合は、ElastiCache for Redis のバージョン で Redis を選択します。

- ElastiCache for Redis バージョン 7.0 (拡張)

[Redis 関数](#)、[シャードされた Pub/Sub](#)、または [Redis ACL の改善](#) を使用したい。詳細については、「[Redis バージョン 7.0 \(拡張\)](#)」を参照してください。

- ElastiCache for Redis バージョン 6.2 (拡張)

r6gd ノードタイプを使用して、メモリと SSD の間でデータを階層化する機能が必要です。詳細については、[データ階層化](#) を参照してください。

- ElastiCache for Redis バージョン 6.0 (拡張)

ロールベースのアクセスコントロールを使用してユーザーを認証します。

詳細については、[Redis バージョン 6.0 \(拡張\)](#) を参照してください。

- ElastiCache for Redis バージョン 5.0.0 (拡張)

プロデューサーが新しいアイテムをリアルタイムで追加できるようにし、コンシューマーがブロッキングまたはノンブロッキングの方法でメッセージを使用できるようにもするログデータ構造である [Redis ストリーム](#) を使用します。

詳細については、「[Redis バージョン 5.0.0 \(拡張\)](#)」を参照してください。

- ElastiCache for Redis バージョン 4.0.10 (拡張)

暗号化および Redis (クラスターモードが有効) クラスターからのシャードの動的な追加または削除をサポートします。

詳細については、「[Redis バージョン 4.0.10 \(拡張\)](#)」を参照してください。

以下のバージョンは廃止か、終了か、間もなく終了します。

- ElastiCache for Redis バージョン 3.2.10 (拡張)

Redis (クラスターモードが有効) クラスターからシャードを動的に追加または削除する機能をサポートします。

⚠ Important

現在 ElastiCache、Redis 3.2.10 では暗号化はサポートされていません。

詳細については、次を参照してください。

- [Redis バージョン 3.2.10 \(拡張\)](#)
- Redis のオンラインリシャードニングのベストプラクティスについては、以下を参照してください。
 - [ベストプラクティス: オンラインリシャードニング](#)
 - [Redis 用のオンラインリシャードニングおよびシャードの再分散 \(クラスターモードが有効\)](#)
- Redis クラスターのスケールリングの詳細については、「[スケールリング](#)」を参照してください。
- ElastiCache for Redis バージョン 3.2.6 (拡張)

以前の Redis バージョンの機能に加えて以下の機能が必要な場合は、Redis 3.2.6 ElastiCache のを選択します。

- 転送時の暗号化 詳細については、「[Amazon ElastiCache for Redis 転送時の暗号化](#)」を参照してください。
- 保管時の暗号化 詳細については、「[Amazon ElastiCache for Redis 保管時の暗号化](#)」を参照してください。
- ElastiCache for Redis (クラスターモードが有効) バージョン 3.2.4

Redis 2.8.x の機能に加え、以下の機能が必要な場合は、Redis 3.2.4 (クラスターモード) を選択します。

- 2 ~ 500 のノードグループ間でデータを分割する必要がある (クラスターモードのみ)。
- 地理空間インデックス作成 (クラスターモードまたは非クラスターモード) が必要。
- 複数のデータベースをサポートする必要がない。
- ElastiCache for Redis (非クラスターモード) 2.8.x および 3.2.4 (拡張)

以下がお客様の状況に当てはまる場合は、Redis 2.8.x または Redis 3.2.4 (非クラスターモード) を選択します。

- 文字列、ハッシュ、リスト、セット、ストアドセット、ビットマップなど、複雑なデータ型が必要である。

- インメモリデータセットをソートまたはランク付けする必要がある。
- キーストアの永続性が必要である。
- 読み取り量が多いアプリケーションのために、プライマリからのデータを 1 つ以上のリードレプリカにレプリケートする必要がある。
- プライマリノードが失敗した場合に、自動的なフェイルオーバーが必要である。
- 発行とサブスクライブ (pub/sub) 機能が必要—クライアントにサーバー上のイベントを通知する必要がある。
- 独自設計型クラスターとサーバーレスキャッシュのバックアップおよび復元機能が必要です。
- 複数のデータベースをサポートする必要がある。

Memcached、Redis (クラスターモードが無効)、Redis (クラスターモードが有効) の比較の概要

	Memcached	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
エンジンバージョン +	1.4.5 以降	4.0.10 以降	4.0.10 以降
データ型	シンプル	2.8.x - 混在 * 複雑	3.2.x 以降 - 複雑
データのパーティション化	はい	いいえ	はい
クラスターが変更可能	はい	はい	3.2.10 以降 - 限定
オンラインリシャーディング	いいえ	いいえ	3.2.10 以降
暗号化	転送中の 1.6.12 以降	4.0.10 以降	4.0.10 以降
データ階層化	いいえ	6.2 以降	6.2 以降
コンプライアンス認定			
コンプライアンス認定			
FedRAMP	はい - 1.6.12 以降	4.0.10 以降	4.0.10 以降
HIPAA	はい - 1.6.12 以降	4.0.10 以降	4.0.10 以降
PCI DSS	はい	4.0.10 以降	4.0.10 以降
マルチスレッド	はい	いいえ	いいえ
ノードタイプのアップグレード	いいえ	はい	はい

	Memcached	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
エンジンのアップグレード	はい	はい	はい
高可用性 (レプリケーション)	いいえ	はい	はい
自動フェイルオーバー	いいえ	オプションです。	必須
パブリック/サブ機能	いいえ	はい	はい
ソートされたセット	いいえ	はい	はい
バックアップと復元	Serverless Memcached のみ、独自設計型 Memcached クラスターは対象外	はい	はい
地理空間インデックス作成	いいえ	4.0.10 以降	はい

注意:

文字列、オブジェクト (データベースなど)

*文字列セット、並べ替えられたセット、リスト、ハッシュ、ビットマップ、Hyperloglog

文字列、セット、ソートされたセット、リスト、ハッシュ、ビットマップ、hyperloglog、地理空間インデックス

+ 非推奨、サポート終了間近、またはサポート終了間近のバージョンを除外します。

クラスターのエンジンを選択した後は、そのエンジンの最新バージョンを使用することをお勧めします。詳細については、[「Memcached バージョン ElastiCache でサポートされる」](#)または[「Redis バージョン ElastiCache でサポートされる」](#)を参照してください。

ElastiCache へのオンライン移行

オンライン移行を使用して、Amazon EC2 のセルフホスト型オープンソース Redis から Amazon ElastiCache にデータを移行できます。

Note

オンライン移行は、r6gd ノードタイプで動作する ElastiCache サーバーレスキャッシュまたはクラスタではサポートされていません。

概要

Amazon EC2 で実行されているオープンソース Redis から Amazon ElastiCache にデータを移行するには、既存または新規作成の Amazon ElastiCache デプロイが必要です。このデプロイの設定は、移行可能な設定になっている必要があります。また、インスタンスのタイプ、シャードの数、レプリカの数などの属性を含め、目的の設定と一致している必要があります。

オンライン移行は、Amazon EC2 上のセルフホスト型オープンソース Redis から Redis for ElastiCache へのデータ移行のために設計されており、Redis for ElastiCache クラスタ間でのものではありません。

Important

オンラインの移行プロセスを開始する前に、以下のセクションをすべて参照することを強くお勧めします。

StartMigration API オペレーションまたは AWS CLI コマンドを呼び出すと、移行が開始されます。Redis クラスタモードが無効になっている場合、移行プロセスでは、ElastiCache for Redis クラスタのプライマリノードがソース Redis プライマリのレプリカになります。Redis クラスタモードが有効になっている場合、移行プロセスにより、各 ElastiCache シャードのプライマリノードが、同じスロットを所有するソースクラスタの対応するシャードのレプリカになります。

クライアント側の変更の準備ができたら、CompleteMigration API オペレーションを呼び出します。この API オペレーションにより、ElastiCache デプロイが、プライマリノードおよびレプリカノードとともに (該当する場合)、プライマリ Redis デプロイに昇格されます。この時点で、クライアントアプリケーションをリダイレクトし、ElastiCache へのデータの書き込みを開始できます。移

行全体を通して、レプリケーションのステータスを確認するには、Redis ノードと ElastiCache プライマリノードで [redis-cli INFO](#) コマンドを実行します。

移行手順

以下のトピックでは、データの移行プロセスの概要を示します。

- [ソースおよびターゲット Redis ノードの移行の準備](#)
- [データ移行のテスト](#)
- [移行の開始](#)
- [データ移行の進行状況の確認](#)
- [データ移行の完了](#)

ソースおよびターゲット Redis ノードの移行の準備

以下に示す 4 つすべての前提条件を満たしていることを確認してから、ElastiCache コンソール、API、または AWS CLI からの移行を開始してください。

ソースおよびターゲット Redis ノードの移行を準備するには

1. ターゲット ElastiCache デプロイを識別し、データの移行先にできることを確認します。

既存または新規作成の ElastiCache デプロイは、移行に関する以下の要件を満たす必要があります。

- Redis エンジンのバージョン 5.0.6 以降を使用している。
- 転送中の暗号化も保管時の暗号化も無効になっている。
- マルチ AZ が有効になっている。
- Redis クラスターからのデータに適合するのに十分なメモリがある。予約メモリを適切に設定するには、「[予約メモリの管理](#)」を参照してください。
- クラスターモードが無効であれば、CLI またはコンソールで CLI または Redis バージョン 5.0.6 以降を使用している場合は Redis バージョン 2.8.21 以降から Redis バージョン 5.0.6 以降に直接移行できます。クラスターモードが有効であれば、CLI またはコンソールで CLI または Redis バージョン 5.0.6 以降を使用している場合はクラスターモードが有効なすべての Redis バージョンから Redis バージョン 5.0.6 以降に直接移行できます。
- ソースとターゲットが一致するシャードの数。
- グローバルデータストアの一部ではありません。

- データ階層化は無効になっています。
2. オープンソースの Redis と、ElastiCache for Redis のデプロイの設定に互換性があることを確認します。

少なくとも、ターゲット ElastiCache デプロイの以下のすべての設定は、Redis レプリケーションの Redis の設定と互換性があることが必要です。

- Redis クラスターで、Redis AUTH が無効になっている必要があります。
 - Redis 設定 `protected-mode` が `no` になっている必要があります。
 - Redis 設定に `bind` 設定が含まれている場合は、ElastiCache ノードへのリクエストを許可するように更新する必要があります。
 - 論理データベースの数は、ElastiCache ノードと、Redis クラスターとで同じである必要があります。この値は、Redis 設定の `databases` を使用して設定します。
 - データに変更を加える Redis コマンドの名前は変更しないでください。変更すると、データのレプリケーションに失敗します (例: `sync`、`psync`、`info`、`config`、`command`、`cluster`)。
 - Redis クラスターからデータを ElastiCache にレプリケートするには、この追加の負荷を処理するのに十分な CPU とメモリがあることが必要です。この負荷は、Redis クラスターによって作成されてネットワークを介して ElastiCache ノードに転送される RDB ファイルから発生します。
 - ソースクラスターのすべての Redis インスタンスは同じポートで実行されている必要があります。
3. 以下を実行して、インスタンスが ElastiCache に接続できることを確認します。
 - 各インスタンスの IP アドレスがプライベートであることを確認します。
 - インスタンスの Redis と同じ仮想プライベートクラウド (VPC) に ElastiCache デプロイを割り当てるか、作成します (推奨)。
 - VPC が異なる場合は、VPC ピア接続を設定して、これらのノード間のアクセスを許可します。VPC ピア接続の詳細については、「[Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)」を参照してください。
 - Redis インスタンスにアタッチされたセキュリティグループは、ElastiCache ノードからのインバウンドトラフィックを許可する必要があります。
 4. データの移行が完了した後、アプリケーションがトラフィックを ElastiCache ノードに転送できることを確認します。詳細については、「[Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)」を参照してください。

データ移行のテスト

すべての前提条件を満たしたら、AWS Management Console、ElastiCache API、または AWS CLI を使用して移行セットアップを検証できます。以下に示しているのは、CLI を使用した例です。

以下のパラメータを指定して `test-migration` コマンドを呼び出し、移行をテストします。

- `--replication-group-id` – データの移行先となるレプリケーショングループの ID。
- `--customer-node-endpoint-list` – データの移行元となるエンドポイントのリスト。リストには要素を 1 つだけ含める必要があります。

以下に示しているのは、CLI を使用した例です。

```
aws elasticache test-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

ElastiCache は実際にデータを移行することなく、移行セットアップを検証します。

移行の開始

すべての前提条件を満たせたら、AWS Management Console、ElastiCache API、または AWS CLI を使用してデータ移行を開始できます。クラスターモードが有効になっている場合、スロットの移行が異なると、ライブ移行の前にリシャーディングが実行されます。以下に示しているのは、CLI を使用した例です。

Note

TestMigration API を使用して移行セットアップを検証することをお勧めします。ただし、これは完全にオプションです。

以下のパラメータを指定して `start-migration` コマンドを呼び出して、移行を開始します。

- `--replication-group-id` – ターゲット ElastiCache レプリケーショングループの識別子。
- `--customer-node-endpoint-list` – エンドポイント (DNS または IP アドレス、およびソース Redis クラスターが実行されているポートも含む) のリスト。このリストには、クラスターモードが無効になっている場合でも、有効になっている場合でも、1 つの要素しか指定できません。連鎖

レプリケーションを有効にしている場合、エンドポイントは Redis クラスターのプライマリノードの代わりにレプリカを参照できます。

以下に示しているのは、CLI を使用した例です。

```
aws elasticache start-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

このコマンドを実行すると、ElastiCache プライマリノードは、各シャードで、(クラスターが有効な Redis の同じスロットを所有する対応するシャードにおいて) Redis インスタンスのレプリカになるようにそれ自体を設定します。ElastiCache クラスターのステータスが [移行中] に変わり、Redis インスタンスから ElastiCache プライマリノードへのデータの移行が開始されます。データのサイズと Redis インスタンスの負荷によっては、移行が完了するまでに時間がかかる場合があります。移行の進行状況を確認するには、Redis インスタンスと ElastiCache プライマリノードで [redis-cli INFO](#) コマンドを実行します。

レプリケーションが正常に完了すると、Redis インスタンスへのすべての書き込みが ElastiCache クラスターに伝播されます。読み取りには ElastiCache ノードを使用できます。ただし、ElastiCache クラスターに書き込むことはできません。ElastiCache プライマリノードに他のレプリカノードが接続されている場合、これらのレプリカノードは ElastiCache プライマリノードからのレプリケートを継続します。このようにして、Redis クラスターのすべてのデータが ElastiCache クラスターのすべてのノードにレプリケートされます。

ElastiCache プライマリノードが Redis インスタンスのレプリカになることができない場合、数回再試行してから最終的に昇格してプライマリに戻ります。ElastiCache クラスターのステータスは [available (使用可能)] に変わり、移行開始の失敗に関するレプリケーショングループイベントが送信されます。このような障害のトラブルシューティングを行うには、以下の点を確認します。

- レプリケーショングループイベントを確認します。イベントからの具体的な情報を使用して、移行の障害を修正します。
- イベントから具体的な情報を得られない場合は、「[ソースおよびターゲット Redis ノードの移行の準備](#)」のガイドラインに従っていることを確認します。
- VPC とサブネットのルーティング設定で、ElastiCache ノードと Redis インスタンスとの間のトラフィックが許可されていることを確認します。
- Redis インスタンスにアタッチされたセキュリティグループが、ElastiCache ノード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。

- レプリケーション固有の障害の詳細については、Redis インスタンスの Redis ログを確認してください。

データ移行の進行状況の確認

データ移行が開始されたら、以下の手順を実行してその進行状況を追跡できます。

- ElastiCache プライマリノードで INFO コマンドを実行して、Redis の `master_link_status` が `up` であることを確認します。この情報は ElastiCache コンソールでも確認できます。クラスターを選択し、[CloudWatch metrics] (CloudWatch メトリクス) で [Primary Link Health Status] (プライマリリンクのヘルスステータス) を確認します。値が 1 に達すると、データは同期されます。
- ElastiCache レプリカが [オンライン] 状態であることを確認するには、Redis インスタンスで INFO コマンドを実行します。これにより、レプリケーション遅延に関する情報も得られます。
- Redis インスタンスで Redis の [CLIENT LIST](#) コマンドを使用して、低クライアント出力バッファを検証します。

データ移行が完了すると、データは、Redis クラスターのプライマリノードへの新しい書き込みと同期されます。

データ移行の完了

ElastiCache クラスターに切り替える準備ができたら、以下のパラメータを指定して `complete-migration` CLI コマンドを使用します。

- `--replication-group-id` – レプリケーショングループの識別子。
- `--force` – データの同期を保たずに移行を強制的に停止するための値。

次に例を示します。

```
aws elasticache complete-migration --replication-group-id test-cluster
```

このコマンドを実行すると、ElastiCache プライマリノードは (各シャードで) Redis インスタンスからのレプリケートを停止し、プライマリに昇格します。通常、この昇格は数分で完了します。プライマリへの昇格を確認するには、イベント Complete Migration successful for test-cluster を確認します。この時点で、アプリケーションに ElastiCache への書き込みと読み取りを指示できます。ElastiCache クラスターの状態は [移行中] から [使用可能] に変わるはずです。

プライマリへの昇格が失敗した場合、ElastiCache プライマリノードは Redis インスタンスからのレプリケートを継続します。ElastiCache クラスターは引き続き [migrating (移行中)] ステータスになり、障害に関するレプリケーショングループイベントメッセージが送信されます。この障害のトラブルシューティングを行うには、以下を参照してください。

- レプリケーショングループイベントを確認します。イベントからの具体的な情報を使用して、障害を修正します。
- 同期していないデータに関するイベントメッセージが表示される場合があります。その場合は、ElastiCache プライマリが Redis インスタンスからレプリケートでき、両方が同期していることを確認します。どうしても移行を停止する必要がある場合は、`-force` オプションを指定して上記のコマンドを実行できます。
- ElastiCache ノードの 1 つが置換中である場合、イベントメッセージが表示されることがあります。置換が完了したら、移行を完了するステップを再試行できます。

コンソールを使用したオンラインデータ移行の実行

AWS Management Console を使用して、クラスターから Redis クラスターにデータを移行できます。

コンソールを使用してオンラインデータ移行を実行するには

1. コンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 新しい Redis クラスターを作成するか、既存のクラスターを選択します。クラスターが以下の要件を満たしていることを確認します。
 - Redis エンジンのバージョンが 5.0.6 以上であることが必要です。
 - Redis クラスターで、Redis AUTH が無効になっている必要があります。
 - Redis 設定 `protected-mode` が `no` になっている必要があります。
 - Redis 設定に `bind` 設定が含まれている場合は、ElastiCache ノードへのリクエストを許可するように更新する必要があります。
 - データベースの数は、ElastiCache ノードと Redis クラスターとで同じである必要があります。この値は、Redis 設定の `databases` を使用して設定します。
 - データに変更を加える Redis コマンドの名前は変更しないことが必要です。変更すると、データのレプリケーションに失敗します。

- Redis クラスターからデータを ElastiCache にレプリケートするには、この追加の負荷を処理するのに十分な CPU とメモリがあることが必要です。この負荷は、Redis クラスターによって作成されてネットワークを介して ElastiCache ノードに転送される RDB ファイルから発生します。
 - クラスターは [available (使用可能)] ステータスであることが必要です。
3. クラスターを選択した状態で、[アクション] で [エンドポイントからのデータの移行] を選択します
 4. [エンドポイントからデータを移行] ダイアログボックスで、IP アドレスと、Redis クラスターが使用可能なポートを入力します。

⚠ Important

IP アドレスは正確であることが必要です。アドレスを誤って入力すると、移行は失敗します。

5. [Start Migration (移行の開始)] を選択します。

クラスターが移行を開始すると、[Modifying (変更中)] ステータスに変わり、次に [Migrating (移行中)] ステータスに変わります。

6. ナビゲーションペインで [Events (イベント)] を選択して、移行の進行状況をモニタリングします。

移行プロセスのどの時点でも、移行を停止できます。これを行うには、クラスターを選択し、[アクション] で [データ移行の停止] を選択します。その後、クラスターは [Available (使用可能)] ステータスになります。

移行が成功すると、クラスターは [Available (使用可能)] ステータスになり、イベントログに以下のように表示されます。

```
Migration operation succeeded for replication group ElastiCacheClusterName.
```

移行が失敗すると、クラスターは [Available (使用可能)] ステータスになり、イベントログに以下のように表示されます。

```
Migration operation failed for replication group ElastiCacheClusterName.
```

リージョンとアベイラビリティーゾーンの選択

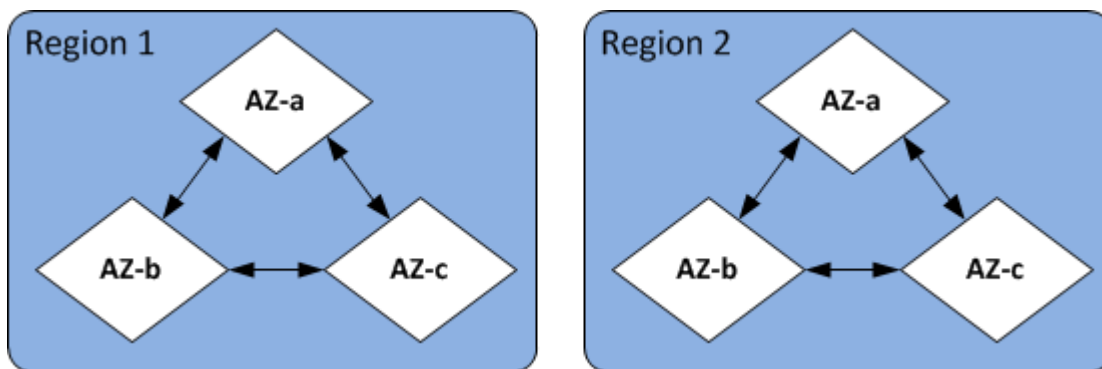
AWS クラウドコンピューティングリソースは、可用性の高いデータセンター施設に收容されています。スケーラビリティと信頼性を向上させるために、これらのデータセンターの設備は物理的に異なる場所に配置されています。これらの場所は、リージョンとアベイラビリティーゾーンに分類されます。

AWS リージョンは大きく、地理的に離れた場所に広く分散しています。アベイラビリティーゾーンは、AWS リージョン内の個別の場所であり、他のアベイラビリティーゾーンの障害から分離されるように設計されています。これらは、同じ AWS リージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供します。

⚠ Important

各リージョンは完全に独立しています。開始した ElastiCache アクティビティ (クラスターの作成など) は、現在のデフォルトリージョンでのみ実行されます。

特定のリージョン内のクラスターを作成または操作するには、対応するリージョンのサービスエンドポイントを使用します。サービスエンドポイントについては、「[サポートされているリージョンおよびエンドポイント](#)」を参照してください。



リージョンとアベイラビリティーゾーン

トピック

- [ノードの配置](#)
- [サポートされているリージョンおよびエンドポイント](#)
- [ElastiCache での Local Zones の使用](#)
- [Outposts の使用](#)

ノードの配置

Amazon ElastiCache では、クラスターのすべてのノードを 1 つまたは複数のアベイラビリティーゾーン (AZs)。さらに、ノードを複数の AZs に配置することを選択した場合 (推奨)、ElastiCache では、ノードごとに AZ を選択するか、ElastiCache で自動的に選択することができます。

ノードを複数の AZ に配置することによって、1 つの AZ での停電などの障害がシステム全体の障害の原因となる可能性を排除できます。1 つの AZ にすべてのノードを配置した場合と、複数の AZ にノードを分散させた場合では、レイテンシーに大きな違いがないことがテストによって証明されました。

クラスターの作成時に各ノードの AZ を指定できます。または、既存のクラスターの変更時にノードを追加して AZ を指定することも可能です。詳細については、次を参照してください。

- [クラスターの作成](#)
- [ElastiCache クラスターの変更](#)
- [クラスターへのノードの追加](#)

サポートされているリージョンおよびエンドポイント

Amazon ElastiCache は複数の AWS リージョンで利用できます。つまり、要件を満たす場所で ElastiCache クラスターを起動できます。例えば、顧客に最も近い AWS リージョンで を起動したり、特定の法的要件を満たすために特定の AWS リージョンで を起動したりできます。

各リージョンは、他のリージョンと完全に分離されるように設計されています。各リージョンには複数のアベイラビリティーゾーン (AZ) があります。ElastiCache サーバーレスキャッシュは、高可用性を実現するために、複数のアベイラビリティーゾーン (ただし us-west-1、データが 2 つのアベイラビリティーゾーンにレプリケートされる を除く) にわたってデータを自動的にレプリケートします。独自の ElastiCache クラスターを設計する場合、さまざまな AZs でノードを起動して耐障害性を実現できます。リージョンとアベイラビリティーゾーンの詳細については、このトピックの最初の「[リージョンとアベイラビリティーゾーンの選択](#)」を参照してください。

ElastiCache がサポートされているリージョン

リージョン名/リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ) リージョン us-east-2	elasticache.us-east-2.amazonaws.com	HTTPS	
米国東部 (バージニア北部) リージョン us-east-1	elasticache.us-east-1.amazonaws.com	HTTPS	
US West (N. California) リージョン us-west-1	elasticache.us-west-1.amazonaws.com	HTTPS	
米国西部 (オレゴン) リージョン us-west-2	elasticache.us-west-2.amazonaws.com	HTTPS	
カナダ (中部) リージョン ca-central-1	elasticache.ca-central-1.amazonaws.com	HTTPS	
カナダ (西部) リージョン ca-west-1	elasticache.ca-west-1.amazonaws.com	HTTPS	
アジアパシフィック (ジャカルタ) ap-southeast-3	elasticache.ap-southeast-3.amazonaws.com	HTTPS	

リージョン名/リージョン	エンドポイント	プロトコル
アジアパシフィック (ムンバイ) リージョン ap-south-1	elasticache.ap-south-1.amazonaws.com	HTTPS
アジアパシフィック (ハイデラバード) リージョン ap-south-2	elasticache.ap-south-2.amazonaws.com	HTTPS
アジアパシフィック (東京) リージョン ap-northeast-1	elasticache.ap-northeast-1.amazonaws.com	HTTPS
アジアパシフィック (ソウル) リージョン ap-northeast-2	elasticache.ap-northeast-2.amazonaws.com	HTTPS
アジアパシフィック (大阪) リージョン ap-northeast-3	elasticache.ap-northeast-3.amazonaws.com	HTTPS
アジアパシフィック (シンガポール) リージョン ap-southeast-1	elasticache.ap-southeast-1.amazonaws.com	HTTPS
アジアパシフィック (シドニー) リージョン ap-southeast-2	elasticache.ap-southeast-2.amazonaws.com	HTTPS

リージョン名/リージョン	エンドポイント	プロトコル	
欧州 (フランクフルト) リージョン eu-central-1	elasticache.eu-central-1.amazonaws.com	HTTPS	
欧州 (チューリッヒ) リージョン eu-central-2	elasticache.eu-central-2.amazonaws.com	HTTPS	
欧州 (ストックホルム) リージョン eu-north-1	elasticache.eu-north-1.amazonaws.com	HTTPS	
中東 (バーレーン) リージョン me-south-1	elasticache.me-south-1.amazonaws.com	HTTPS	
中東 (アラブ首長国連邦) リージョン me-central-1	elasticache.me-central-1.amazonaws.com	HTTPS	
欧州 (アイルランド) リージョン eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
欧州 (ロンドン) リージョン eu-west-2	elasticache.eu-west-2.amazonaws.com	HTTPS	

リージョン名/リージョン	エンドポイント	プロトコル	
欧州 (パリ) リージョン eu-west-3	elasticache.eu-west-3.amazonaws.com	HTTPS	
欧州 (ミラノ) リージョン eu-south-1	elasticache.eu-south-1.amazonaws.com	HTTPS	
欧州 (スペイン) リージョン eu-south-2	elasticache.eu-south-2.amazonaws.com	HTTPS	
南米 (サンパウロ) リージョン sa-east-1	elasticache.sa-east-1.amazonaws.com	HTTPS	
中国 (北京) リージョン cn-north-1	elasticache.cn-north-1.amazonaws.com.cn	HTTPS	
中国 (寧夏) リージョン cn-northwest-1	elasticache.cn-northwest-1.amazonaws.com.cn	HTTPS	
アジアパシフィック (香港) リージョン ap-east-1	elasticache.ap-east-1.amazonaws.com	HTTPS	

リージョン名/リージョン	エンドポイント	プロトコル
アフリカ (ケープタウン) リージョン af-south-1	elasticache.af-south-1.amazonaws.com	HTTPS
イスラエル (テルアビブ) リージョン il-central-1	elasticache.il-central-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部) us-gov-west-1	elasticache.us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud (米国東部) us-gov-east-1	elasticache.us-gov-east-1.amazonaws.com	HTTPS

で AWS GovCloud (米国) を使用する方法については ElastiCache、[AWS GovCloud 「\(米国\) リージョンの サービス ElastiCache」](#) を参照してください。

一部のリージョンでは、ノードタイプのサブセットがサポートされています。AWS リージョン別のサポートされているノードタイプの表については、「」を参照してください[AWS リージョン別のサポート対象ノードタイプ](#)。

リージョン別の AWS 製品とサービスの表については、「[リージョン別の製品とサービス](#)」を参照してください。

ElastiCache での Local Zones の使用

ローカルゾーンは、ユーザーに地理的に近い、AWS リージョンの拡張です。新しいサブネットを作成し、これを Local Zones に割り当てることで、親の AWS リージョンから Local Zones に VPC を

拡張できます。ローカルゾーンにサブネットを作成すると、VPC はそのローカルゾーンに拡張されます。ローカルゾーンのサブネットは、VPC 内の他のサブネットと同じように動作します。

Local Zones を使用すると、ElastiCache クラスターなどのリソースをユーザーに近い複数の場所に配置できます。

ElastiCache クラスターを作成するときに、ローカルゾーンのサブネットを選択できます。Local Zones は、インターネットへの独自の接続を持ち、AWS Direct Connect をサポートします。したがって、ローカルゾーンで作成したリソースは、非常に低いレイテンシーの通信をローカルユーザーに提供できます。詳細については、「[AWS Local Zones](#)」を参照してください。

ローカルゾーンを表すには、AWS リージョンコードに続けて場所を示す識別子を使用します (例: us-west-2-lax-1a)。

現時点では、利用可能な Local Zones は us-west-2-lax-1a と us-west-2-lax-1b です。

Local Zones の ElastiCache には、以下の制限が適用されます。

- グローバルデータストアはサポートされていません。
- オンライン移行はサポートされていません。
- 現時点では、Local Zones では、以下のノードがサポートされています。
 - 現行世代:

M5 ノードタイプ:

cache.m5.large、cache.m5.xlarge、cache.m5.2xlarge、cache.m5.4xlarge、cache.m5.

R5 ノードタイプ:

cache.r5.large、cache.r5.xlarge、cache.r5.2xlarge、cache.r5.4xlarge、cache.r5.

T3 ノードタイプ: cache.t3.micro、cache.t3.small、cache.t3.medium

ローカルゾーンの有効化

1. Amazon EC2 コンソールでローカルゾーンを有効にします。

詳細については、Amazon EC2 ユーザーガイドの「[Local Zones の有効化](#)」を参照してください。

2. ローカルゾーン内にサブネットを作成します。

詳細については、Amazon VPC ユーザーガイドの「[VPC でサブネットを作成する](#)」を参照してください。

- ローカルゾーン内に ElastiCache サブネットグループを作成します。

ElastiCache サブネットグループを作成するときに、ローカルゾーンのアベイラビリティゾーングループを選択します。

詳細については、ElastiCache ユーザーガイドの「[サブネットグループの作成](#)」を参照してください。

- ローカルゾーンで ElastiCache サブネットを使用する ElastiCache for Redis クラスターを作成します。詳細については、次のトピックのいずれかを参照してください。

- [Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)

Outposts の使用

AWS Outposts は、AWS インフラストラクチャ、サービス、APIs、ツールをお客様のオンプレミスに拡張するフルマネージドサービスです。AWS マネージドインフラストラクチャへのローカルアクセスを提供することで、AWS Outposts は AWS、リージョンと同じプログラミングインターフェイスを使用してオンプレミスでアプリケーションを構築および実行し、ローカルコンピューティングとストレージリソースを使用してレイテンシーを短縮し、ローカルデータ処理のニーズを低く抑えることができます。Outpost は、お客様のサイトにデプロイされた AWS コンピューティングおよびストレージ容量のプールです。は、この容量を AWS リージョンの一部として AWS 運用、モニタリング、管理します。Outpost にサブネットを作成し、ElastiCache クラスターなどの AWS リソースを作成するときに指定できます。

Note

このバージョンでは、次のような制限が適用されます。

- ElastiCache for Outposts は M5 および R5 ノードファミリーのみをサポートします。
- マルチ AZ (クロス Outpost レプリケーションはサポートされていません)。
- ライブ移行はサポートされていません。
- ローカルスナップショットはサポートされていません。
- エンジンログとスローログは有効にできません。

- ElastiCache Outposts のは ColP をサポートしていません。
- ElastiCache for Outposts は、cn-north-1、cn-northwest-1、ap-northeast-3 のリージョンではサポートされていません。

Redis コンソールでの Outposts の使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、Redis キャッシュ を選択します。
3. Redis キャッシュの作成 を選択します。
4. クラスター設定 で、独自のキャッシュの設計 とクラスターキャッシュ を選択します。クラスターモードは無効 のままにします。次に、キャッシュの名前とオプションの説明を作成します。
5. ロケーション で、オンプレミス を選択します。
6. 「オンプレミス」セクションに、「Outpost ID」というフィールドが表示されます。クラスターを実行する ID を入力します。

クラスター設定のその他の設定はすべてデフォルトのままにできます。

7. 接続 で、新しいサブネットグループの作成 を選択し、VPC ID を入力します。残りはデフォルトのままにして、次へ を選択します。

オンプレミスオプションを設定する

利用可能な Outpost を選択してキャッシュクラスターを選択するか、利用可能な Outposts がない場合は、次の手順を使用して新しい Outpost を作成できます。

[オンプレミスのオプション] の下で:

1. [Redis の設定] の下で:
 - a. [名前]: Redis クラスターの名前を入力します。
 - b. [説明]: Redis クラスターの説明を入力します。
 - c. エンジンバージョンの互換性 : エンジンバージョンは Outpost AWS リージョンに基づいています

- d. [ポート]: デフォルトポート (6379) のままにしておきます。理由があつて異なるポートを使用する場合は、そのポート番号を入力します。
- e. [パラメータグループ]: ドロップダウンを使用して、デフォルトまたはカスタムパラメータグループを選択します。
- f. [ノードタイプ]: 使用可能なインスタンスは、Outposts 可用性に基づきます。Outposts 用 Porting Assistant for .NET は、M5 および R5 ノードファミリーのみをサポートします。ドロップダウンリストから [Outposts] を選択してから、このクラスターで使用する利用可能なノードタイプを選択します。次に、[保存] を選択します。
- g. [レプリカの数]: このレプリケーショングループ用に作成するリードレプリカの数を入力します。リードレプリカ数は少なくとも 1 個で、最大 5 個です。デフォルト値は 2 です。

自動生成されたリードレプリカの名前は、プライマリクラスターの名前と同じパターンに従います。最後にダッシュと 3 桁の連番が追加され、-002 で開始されます。たとえば、レプリケーショングループの名前が MyGroup の場合、セカンダリの名前は MyGroup-002、MyGroup-003、MyGroup-004、MyGroup-005、MyGroup-006 となります。

2. 接続の下:

- a. [サブネットグループ]: リストから [新規作成] を選択します。
 - [名前] - サブネットグループの名前を入力します。
 - [説明] サブネットグループの説明を入力します。
 - [VPC ID]: VPC ID は Outpost VPC と一致する必要があります。Outposts にサブネット ID がない VPC を選択した場合、リストは空を返します。
 - [アベイラビリティゾーンまたは Outpost]: 使用している Outpost を選択します。
 - [サブネット ID]: Outpost で使用できるサブネット ID を選択します。使用可能なサブネット ID がない場合は、それらのサブネット ID を作成する必要があります。詳細については、「[サブネットの作成](#)」を参照してください。
- b. [作成] を選択します。

Outpost クラスターの詳細の表示

Redis リストページで、AWS Outpost に属するクラスターを選択し、クラスターの詳細を表示するときに次の点に注意してください。

- **アベイラビリティゾーン** : これは、ARN (Amazon リソースネーム) と AWS リソース番号を使用して Outpost を表します。
- **Outpost 名**: Outpost AWS の名前。

AWS CLI での Outposts の使用

AWS Command Line Interface (AWS CLI) を使用して、コマンドラインから複数の AWS サービスを制御したり、スクリプトを使用して自動化したりできます。AWS CLI はアドホック (ワンタイム) オペレーションに使用できます。

のダウンロードと設定 AWS CLI

は Windows、macOS で AWS CLI 実行されます。これをダウンロードして設定するには、次の手順に従います。

CLI をダウンロード、インストール、設定するには

1. [AWS コマンドラインインターフェイス](#) のウェブページで AWS CLI をダウンロードします。
2. ユーザーガイドの「[AWS CLI のインストール](#)」および「[AWS CLI の設定](#)」の手順に従います。AWS Command Line Interface

Outposts での AWS CLI の使用

次の CLI オペレーションを使用して、Outposts を使用するキャッシュクラスターを作成します。

- [create-cache-cluster](#) – このオペレーションを使用して、`outpost-mode` パラメータは、キャッシュクラスター内のノードが単一の Outpost に作成されるか、複数の Outposts にまたがって作成されるかを指定する値を受け入れます。

Note

現時点では、`single-outpost` モードがサポートされています。

```
aws elasticache create-cache-cluster \  
  --cache-cluster-id cache cluster id \  
  --outpost-mode single-outpost \  
  \
```

の使用 ElastiCache

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

トピック

- [スナップショットおよび復元](#)
- [エンジンバージョンとアップグレード](#)
- [ElastiCache ベストプラクティスとキャッシュ戦略](#)
- [独自設計型クラスターの管理](#)
- [Redis ElastiCache のスケーリング](#)
- [ElastiCache for Redis での JSON の使用開始](#)
- [ElastiCache リソースのタグ付け](#)
- [Amazon ElastiCache Well-Architected レンズの使用](#)
- [一般的なトラブルシューティング手順とベストプラクティス](#)
- [その他のトラブルシューティング手順](#)

スナップショットおよび復元

Redis を実行している Amazon ElastiCache キャッシュは、スナップショットを作成してデータをバックアップできます。このバックアップを使用すると、キャッシュまたはシードデータを新しいキャッシュに復元できます。バックアップは、キャッシュ内の全データとキャッシュのメタデータで構成されます。すべてのバックアップは、耐久性のあるストレージを提供する Amazon Simple Storage Service (Amazon S3) に書き込まれます。いつでも、新しい Redis キャッシュを作成し、バックアップからのデータを入力してデータを復元できます。では ElastiCache、AWS Command Line Interface (AWS CLI)、AWS Management Console、および ElastiCache API を使用してバックアップを管理できます。

キャッシュを削除する予定であり、データを保持することが重要な場合は、万が一に備えることができます。そのためには、まず手動バックアップを作成し、そのステータスが [利用可能] であることを確認してから、キャッシュを削除します。これにより、バックアップが失敗した場合でも、キャッシュデータは引き続き使用できます。前述のベストプラクティスに従って、バックアップの作成を再試行できます。

トピック

- [バックアップの制約](#)
- [独自設計型クラスタのバックアップがパフォーマンスに与える影響](#)
- [自動バックアップのスケジュール](#)
- [手動バックアップの取得](#)
- [最終バックアップの作成](#)
- [バックアップの詳細の表示](#)
- [バックアップのコピー](#)
- [バックアップのエクスポート](#)
- [バックアップから新しいキャッシュへの復元](#)
- [バックアップの削除](#)
- [バックアップへのタグ付け](#)
- [外部で作成されたバックアップによる新しい独自設計型クラスタのシード](#)

バックアップの制約

バックアップを計画または作成するときは、以下の制約事項を考慮してください。

- バックアップと復元は、Redis または Serverless Memcached で実行されているキャッシュでのみサポートされます。
- Redis (クラスタモードが無効) クラスタでは、バックアップと復元は `cache.t1.micro` ノードではサポートされません。他のキャッシュノードタイプはすべてサポートされます。
- Redis (クラスタモードが有効) クラスタでは、バックアップ、および復元はすべてのノードタイプでサポートされます。
- 連続する 24 時間の間は、サーバーレスキャッシュごとに最大 24 個の手動バックアップを作成できます。Redis 独自設計型クラスタの場合、クラスタ内のノードごとに作成できる手動バックアップは 20 個までです。
- Redis (クラスタモードが有効) は、クラスタレベル (API または CLI ではレプリケーショングループレベル) でのバックアップの作成のみをサポートします。Redis (クラスタモードが有効) は、シャードレベル (API または CLI ではノードグループレベル) でのバックアップの作成をサポートしていません。
- バックアッププロセス中に、サーバーレスキャッシュで他の API または CLI オペレーションを実行することはできません。バックアップ中に独自設計型クラスタで API または CLI オペレーションを実行できます。

- データ階層化でキャッシュを使用する場合、バックアップを Amazon S3 にエクスポートすることはできません。
- r6gd ノードタイプを使用するクラスターのバックアップは、r6gd ノードタイプを使用するクラスターにのみ復元できます。

独自設計型クラスターのバックアップがパフォーマンスに与える影響

サーバーレスキャッシュのバックアップはアプリケーションに対して透過的であり、パフォーマンスには影響しません。ただし、独自設計型クラスターのバックアップを作成する場合、使用可能な予約メモリによっては、パフォーマンスに影響を及ぼす可能性があります。独自設計型クラスターは、ElastiCache および Memcached では使用できませんが、ElastiCache および Redis で使用できます。

以下に示しているのは、独自設計型クラスターのバックアップパフォーマンスを向上させるためのガイドラインです。

- reserved-memory-percent パラメータを設定する – 過剰なページングを軽減するため、reserved-memory-percent パラメータを設定することをお勧めします。このパラメータは、Redis がノードの使用可能なメモリをすべて消費することを防止し、ページング容量を削減するのに役立ちます。また、大容量のノードを使用するだけでパフォーマンスが向上する場合があります。reserved-memory と reserved-memory-percent パラメータの詳細については、「[予約メモリの管理](#)」を参照してください。
- リードレプリカからバックアップを作成する – 複数のノードを含むノードグループで Redis を実行している場合、プライマリノードまたはいずれかのリードレプリカからバックアップを作成できます。BGSAVE の実行時に必要なシステムリソースのために、いずれかのリードレプリカからバックアップを作成することをお勧めします。レプリカからバックアップが作成されている間、プライマリノードは BGSAVE リソースの要件の影響を受けません。プライマリノードは、速度を落とすことなくリクエストを処理し続けることができます。

これを行うには、[手動バックアップの作成](#) を参照し、[バックアップの作成] ウィンドウの [クラスター名] フィールドで、デフォルトのプライマリノードではなくレプリカを選択します。

レプリケーショングループを削除して最終バックアップをリクエストする場合、ElastiCache は常にプライマリノードからバックアップを取得します。これにより、レプリケーショングループが削除される前に、最新の Redis データがキャプチャされます。

自動バックアップのスケジュール

Redis サーバーレスキャッシュまたは独自設計型クラスターの自動バックアップを有効にできます。自動バックアップが有効になっている場合、はキャッシュのバックアップを毎日 ElastiCache 作成します。キャッシュへの影響はなく、変更は即時に行われます。自動バックアップは、データ損失を防ぐのに役立ちます。障害が起こった場合、最新のバックアップからデータを復元して新しいキャッシュを作成できます。その結果、データがプリロードされたキャッシュがウォームスタートされ、使用可能になります。詳細については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

自動バックアップをスケジュールする場合は、次の設定を検討する必要があります:

- **バックアップ開始時刻** — がバックアップの作成 ElastiCache を開始する時刻。バックアップ期間は、最も便利な時間に設定できます。バックアップウィンドウを指定しない場合、は自動的にバックアップウィンドウを ElastiCache 割り当てます。
- **[バックアップ保持期限]** – バックアップが Amazon S3 に保持される日数。たとえば、保持期限を 5 に設定すると、今日作成されたバックアップは 5 日間保持されます。保持期限が切れると、バックアップは自動的に削除されます。

最大バックアップ保持期限は 35 日です。バックアップ保持期限を 0 に設定すると、キャッシュの自動バックアップが無効になります。

ElastiCache コンソール、または ElastiCache API を使用して、新しいキャッシュを作成するとき、または既存の Redis キャッシュを更新するときに、自動バックアップを有効 AWS CLI または無効にできます。これは、詳細 Redis 設定セクションの「自動バックアップを有効にする」ボックスをチェックすることによって行われます。

手動バックアップの取得

自動バックアップに加えて、いつでも手動バックアップを作成できます。指定された保持期間後に自動的に削除される自動バックアップとは異なり、手動バックアップには、経過した後で自動的に削除される保持期間はありません。キャッシュを削除しても、そのキャッシュからの手動バックアップはすべて保持されます。手動バックアップを保持する必要がなくなった場合は、自分で明示的に削除する必要があります。

手動バックアップの直接的な作成に加えて、以下のいずれかの方法で手動バックアップを作成できます。

- 「[バックアップのコピー](#)」ソースのバックアップが自動で作成されたか、手動で作成されたかは問題ではありません。
- 「[最終バックアップの作成](#)」クラスターまたはノードを削除する直前に最終バックアップを作成します。

キャッシュの手動バックアップは、AWS Management Console、AWS CLIまたはElastiCache APIを使用して作成できます。

手動バックアップの作成

キャッシュのバックアップを作成するには (コンソール)

1. 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。
2. ナビゲーションペインで、[Redis キャッシュ] を選択します。
3. バックアップするキャッシュの名前の左側にあるボックスを選択します。
4. [バックアップ] を選択します。
5. [バックアップを作成する] ダイアログで、[バックアップ名] ボックスにバックアップの名前を入力します。バックアップ元のクラスターおよびバックアップを実行した日付と時刻を示すような名前にすることをお勧めします。

クラスターの命名に関する制約は次のとおりです。

- 1~40 個の英数字またはハイフンを使用する必要があります。
- 先頭は文字を使用する必要があります。

- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。

6. [バックアップを作成] を選択します。

クラスターのステータスが `snapshotting` に変わります。

手動バックアップの作成 (AWS CLI)

を使用したサーバーレスキャッシュの手動バックアップ AWS CLI

を使用してキャッシュの手動バックアップを作成するには AWS CLI、以下のパラメータを指定して `create-serverless-snapshot` AWS CLI オペレーションを使用します。

- `--serverless-cache-name` — バックアップするサーバーレスキャッシュの名前。
- `--serverless-cache-snapshot-name` – 作成するスナップショットの名前。

Linux、macOS、Unix の場合:

- ```
aws elasticache create-serverless-snapshot \
 --serverless-cache-name CacheName \
 --serverless-cache-snapshot-name bkup-20231127
```

Windows の場合:

- ```
aws elasticache create-serverless-snapshot ^  
    --serverless-cache-name CacheName ^  
    --serverless-cache-snapshot-name bkup-20231127
```

を使用した独自設計型クラスターの手動バックアップ AWS CLI

を使用して独自設計型クラスターの手動バックアップを作成するには AWS CLI、以下のパラメータを指定して `create-snapshot` AWS CLI オペレーションを使用します。

- `--cache-cluster-id`
 - バックアップするクラスターにレプリカノードがない場合、`--cache-cluster-id` は、バックアップするクラスターの名前 (*mycluster* など) です。

- バックアップするクラスターに 1 つ以上のレプリカノードがある場合、`--cache-cluster-id` は、バックアップに使用するクラスター内のノードの名前です。たとえば、名前は `mycluster-002` です。

このパラメータは、Redis (クラスターモードが無効) クラスターをバックアップする場合にのみ使用します。

- `--replication-group-id` – バックアップのソースとして使用する Redis (クラスターモードが有効) クラスター (CLI/API: レプリケーショングループ) の名前。このパラメータは、Redis (クラスターモードが有効) クラスターをバックアップするときに使用します。

- `--snapshot-name` – 作成するスナップショットの名前。

クラスターの命名に関する制約は次のとおりです。

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

例 1: レプリカノードがない Redis (クラスターモードが無効) クラスターのバックアップ

次の AWS CLI オペレーションでは、リードレプリカ `myNonClusteredRedis` がない Redis (クラスターモードが無効) クラスター `bkup-20150515` からバックアップを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-snapshot \  
  --cache-cluster-id myNonClusteredRedis \  
  --snapshot-name bkup-20150515
```

Windows の場合:

```
aws elasticache create-snapshot ^  
  --cache-cluster-id myNonClusteredRedis ^  
  --snapshot-name bkup-20150515
```


例 2: レプリカノードがある Redis (クラスターモードが無効) クラスターのバックアップ

次の AWS CLI オペレーションでは、Redis (クラスターモードが無効) クラスター `bkup-20150515` からバックアップを作成します `myNonClusteredRedis`。このバックアップには、1 つ以上のリードレプリカがあります。

Linux、macOS、Unix の場合:

```
aws elasticache create-snapshot \  
  --cache-cluster-id myNonClusteredRedis-001 \  
  --snapshot-name bkup-20150515
```

Windows の場合:

```
aws elasticache create-snapshot ^  
  --cache-cluster-id myNonClusteredRedis-001 ^  
  --snapshot-name bkup-20150515
```

出力の例: レプリカノードがある Redis (クラスターモードが無効) クラスターのバックアップ

このオペレーションによる出力は以下のようになります。

```
{  
  "Snapshot": {  
    "Engine": "redis",  
    "CacheParameterGroupName": "default.redis6.x",  
    "VpcId": "vpc-91280df6",  
    "CacheClusterId": "myNonClusteredRedis-001",  
    "SnapshotRetentionLimit": 0,  
    "NumCacheNodes": 1,  
    "SnapshotName": "bkup-20150515",  
    "CacheClusterCreateTime": "2017-01-12T18:59:48.048Z",  
    "AutoMinorVersionUpgrade": true,  
    "PreferredAvailabilityZone": "us-east-1c",  
    "SnapshotStatus": "creating",  
    "SnapshotSource": "manual",  
    "SnapshotWindow": "08:30-09:30",  
    "EngineVersion": "6.0",  
    "NodeSnapshots": [  
      {  
        "CacheSize": "",  
        "CacheNodeId": "0001",
```

```
        "CacheNodeCreateTime": "2017-01-12T18:59:48.048Z"
      }
    ],
    "CacheSubnetGroupName": "default",
    "Port": 6379,
    "PreferredMaintenanceWindow": "wed:07:30-wed:08:30",
    "CacheNodeType": "cache.m3.2xlarge",
    "DataTiering": "disabled"
  }
}
```

例 3: Redis (クラスターモードが有効) クラスターのバックアップ

次の AWS CLI オペレーションでは、Redis (クラスターモードが有効) クラスター bkup-20150515 からバックアップを作成します myClusteredRedis。送信元を特定する --cache-cluster-id ではなく、--replication-group-id を使用することに注意してください。

Linux、macOS、Unix の場合:

```
aws elasticache create-snapshot \  
  --replication-group-id myClusteredRedis \  
  --snapshot-name bkup-20150515
```

Windows の場合:

```
aws elasticache create-snapshot ^  
  --replication-group-id myClusteredRedis ^  
  --snapshot-name bkup-20150515
```

出力の例: Redis (クラスターモードが有効) クラスターのバックアップ

このオペレーションによる出力は、次のようになります。

```
{  
  "Snapshot": {  
    "Engine": "redis",  
    "CacheParameterGroupName": "default.redis6.x.cluster.on",  
    "VpcId": "vpc-91280df6",  
    "NodeSnapshots": [  
      {
```

```
        "CacheSize": "",
        "NodeGroupId": "0001"
    },
    {
        "CacheSize": "",
        "NodeGroupId": "0002"
    }
],
"NumNodeGroups": 2,
"SnapshotName": "bkup-20150515",
"ReplicationGroupId": "myClusteredRedis",
"AutoMinorVersionUpgrade": true,
"SnapshotRetentionLimit": 1,
"AutomaticFailover": "enabled",
"SnapshotStatus": "creating",
"SnapshotSource": "manual",
"SnapshotWindow": "10:00-11:00",
"EngineVersion": "6.0",
"CacheSubnetGroupName": "default",
"ReplicationGroupDescription": "2 shards 2 nodes each",
"Port": 6379,
"PreferredMaintenanceWindow": "sat:03:30-sat:04:30",
"CacheNodeType": "cache.r3.large",
"DataTiering": "disabled"
}
}
```

関連トピック

詳細については、AWS CLI コマンドリファレンスの「[create-snapshot](#)」を参照してください。

最終バックアップの作成

コンソール、ElastiCache、または ElastiCache API を使用して AWS CLI 最終バックアップを作成できます。

最終バックアップの作成 (コンソール)

ElastiCache コンソールを使用して Redis サーバーレスキャッシュまたは独自設計型クラスターを削除するときに、最終バックアップを作成できます。

キャッシュの削除時に最終バックアップを作成するには、削除ダイアログボックスの「バックアップの作成」で「はい」を選択し、バックアップに名前を付けます。

関連トピック

- [AWS Management Console を使用する場合](#)
- [レプリケーショングループの削除 \(コンソール\)](#)

最終バックアップの作成 (AWS CLI)

を使用してキャッシュを削除するときに、最終バックアップを作成できます AWS CLI。

トピック

- [サーバーレスキャッシュを削除する場合](#)
- [リードレプリカなしで Redis 独自設計型クラスターを削除する場合](#)
- [リードレプリカのある Redis クラスターを削除する場合](#)

サーバーレスキャッシュを削除する場合

最終バックアップを作成するには、以下のパラメータを指定して `delete-serverless-cache` AWS CLI オペレーションを使用します。

- `--serverless-cache-name` – 削除するキャッシュの名前。
- `--final-snapshot-name` – バックアップの名前。

以下のコードは、最終バックアップ `bkup-20231127-final` をキャッシュ `myserverlesscache` の削除時に作成します。

Linux、macOS、Unix の場合:

```
aws elasticache delete-serverless-cache \  
  --serverless-cache-name myserverlesscache \  
  --final-snapshot-name bkup-20231127-final
```

Windows の場合:

```
aws elasticache delete-serverless-cache ^  
  --serverless-cache-name myserverlesscache ^  
  --final-snapshot-name bkup-20231127-final
```

詳細については、AWS CLI コマンドリファレンスの「[delete-serverless-cache](#)」を参照してください。

リードレプリカなしで Redis 独自設計型クラスターを削除する場合

リードレプリカのない独自設計型クラスターの最終バックアップを作成するには、以下のパラメータを指定して `delete-cache-cluster` AWS CLI オペレーションを使用します。

- `--cache-cluster-id` – 削除するクラスターの名前。
- `--final-snapshot-identifier` – バックアップの名前。

以下のコードは、最終バックアップ `bkup-20150515-final` をクラスター `myRedisCluster` の削除時に作成します。

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-cluster \  
  --cache-cluster-id myRedisCluster \  
  --final-snapshot-identifier bkup-20150515-final
```

Windows の場合:

```
aws elasticache delete-cache-cluster ^  
  --cache-cluster-id myRedisCluster ^  
  --final-snapshot-identifier bkup-20150515-final
```

詳細については、AWS CLI コマンドリファレンスの「[delete-cache-cluster](#)」を参照してください。

リードレプリカのある Redis クラスターを削除する場合

レプリケーショングループを削除するときに最終バックアップを作成するには、以下のパラメータを指定して `delete-replication-group` AWS CLI オペレーションを使用します。

- `--replication-group-id` – 削除するレプリケーショングループの名前。
- `--final-snapshot-identifier` – 最終バックアップの名前。

以下のコードは、最終バックアップ `bkup-20150515-final` をレプリケーショングループ `myReplGroup` の削除時に作成します。

Linux、macOS、Unix の場合:

```
aws elasticache delete-replication-group \  
    --replication-group-id myReplGroup \  
    --final-snapshot-identifier bkup-20150515-final
```

Windows の場合:

```
aws elasticache delete-replication-group ^  
    --replication-group-id myReplGroup ^  
    --final-snapshot-identifier bkup-20150515-final
```

詳細については、AWS CLI コマンドリファレンスの「[delete-replication-group](#)」を参照してください。

バックアップの詳細の表示

以下の手順では、バックアップのリストを表示する方法を示しています。必要に応じて、特定のバックアップの詳細を表示することもできます。

バックアップの説明 (コンソール)

を使用してバックアップを表示するには AWS Management Console

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで [バックアップ] を選択します。
3. 特定のバックアップの詳細を表示するには、バックアップの名前の左にあるチェックボックスをオンにします。

サーバーレスバックアップの説明 (AWS CLI)

サーバーレスバックアップのリストと必要に応じて特定のバックアップの詳細を表示するには、`describe-serverless-cache-snapshots` CLI オペレーションを使用します。

例

以下のオペレーションでは、パラメータ `--max-records` を使用して、アカウントに関連付けられた最大 20 個のバックアップをリスト表示します。パラメータ `--max-records` を省略すると、最大 50 個のバックアップが一覧表示されます。

```
aws elasticache describe-serverless-cache-snapshots --max-records 20
```

以下のオペレーションでは、パラメータ `--serverless-cache-name` を使用して、キャッシュ `my-cache` に関連付けられたバックアップのみをリスト表示します。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-name my-cache
```

以下のオペレーションでは、パラメータ `--serverless-cache-snapshot-name` を使用して、バックアップ `my-backup` の詳細を表示します。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-snapshot-name my-backup
```

詳細については、AWS CLI コマンドリファレンスの「[describe-serverless-cache-snapshots](#)」を参照してください。

独自設計型クラスターバックアップの説明 (AWS CLI)

独自設計型クラスターのバックアップのリストと必要に応じて特定のバックアップの詳細を表示するには、describe-snapshots CLI オペレーションを使用します。

例

以下のオペレーションでは、パラメータ `--max-records` を使用して、アカウントに関連付けられた最大 20 個のバックアップをリスト表示します。パラメータ `--max-records` を省略すると、最大 50 個のバックアップが一覧表示されます。

```
aws elasticache describe-snapshots --max-records 20
```

以下のオペレーションでは、パラメータ `--cache-cluster-id` を使用して、クラスター `my-cluster` に関連付けられたバックアップのみをリスト表示します。

```
aws elasticache describe-snapshots --cache-cluster-id my-cluster
```

以下のオペレーションでは、パラメータ `--snapshot-name` を使用して、バックアップ `my-backup` の詳細を表示します。

```
aws elasticache describe-snapshots --snapshot-name my-backup
```

詳細については、コマンドリファレンスの「[describe-snapshots](#)」を参照してください。AWS CLI

バックアップのコピー

自動で作成されたか手動で作成されたかにかかわらず、どのバックアップのコピーでも作成できます。バックアップをエクスポートして、の外部からアクセスすることもできます ElastiCache。バックアップのエクスポートに関するガイダンスについては、「」を参照してください [バックアップのエクスポート](#)。

以下の手順では、バックアップをコピーする方法を示しています。

バックアップのコピー (コンソール)

バックアップをコピーするには (コンソール)

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. バックアップのリストを表示するには、左のナビゲーションペインから [Backups] を選択します。
3. バックアップのリストから、コピーするバックアップの名前の左にあるチェックボックスをオンにします。
4. [アクション]、[コピー] の順に選択します。
5. [新しいバックアップ名] ボックスに、新しいバックアップの名前を入力します。
6. [コピー] を選択します。

サーバーレスバックアップのコピー (AWS CLI)

サーバーレスキャッシュのバックアップをコピーするには、`copy-serverless-cache-snapshot` オペレーションを使用します。

パラメータ

- `--source-serverless-cache-snapshot-name` – コピーするバックアップの名前。
- `--target-serverless-cache-snapshot-name` – バックアップのコピーの名前。

以下の例では、自動バックアップのコピーを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache copy-serverless-cache-snapshot \
```

```
--source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 \  
--target-serverless-cache-snapshot-name my-backup-copy
```

Windows の場合:

```
aws elasticache copy-serverless-cache-snapshot ^  
--source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 ^  
--target-serverless-cache-snapshot-name my-backup-copy
```

詳細については、「AWS CLI」の「[copy-serverless-cache-snapshot](#)」を参照してください。

独自設計型クラスターバックアップのコピー (AWS CLI)

独自設計型クラスターのバックアップをコピーするには、copy-snapshot オペレーションを使用します。

パラメータ

- --source-snapshot-name – コピーするバックアップの名前。
- --target-snapshot-name – バックアップのコピーの名前。
- --target-bucket – バックアップのエクスポート用に予約されています。バックアップのコピーを作成する場合は、このパラメータを使用しないでください。Redis サーバーレスキャッシュと Redis 独自設計型クラスターでのみ使用できます。詳細については、「[バックアップのエクスポート](#)」を参照してください。

以下の例では、自動バックアップのコピーを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache copy-snapshot \  
--source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \  
--target-snapshot-name my-backup-copy
```

Windows の場合:

```
aws elasticache copy-snapshot ^  
--source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 ^  
--target-snapshot-name my-backup-copy
```

詳細については、「AWS CLI」の「[copy-snapshot](#)」を参照してください。

バックアップのエクスポート

Amazon ElastiCache は ElastiCache 、 for Redis バックアップを Amazon Simple Storage Service (Amazon S3) バケットにエクスポートすることをサポートしています。これにより、 の外部からアクセスできるようになります ElastiCache。 ElastiCache コンソール、 、 AWS CLI または ElastiCache API を使用してバックアップをエクスポートできます。

バックアップのエクスポートは、別の AWS リージョンでクラスターを起動する必要がある場合に役立ちます。データを 1 つの AWS リージョンにエクスポートし、.rdb ファイルを新しい AWS リージョンにコピーしてから、その .rdb ファイルを使用して新しいクラスターが使用を通じて入力されるのを待つ代わりに、新しいキャッシュをシードできます。新しいクラスターのシードについては、「[外部で作成されたバックアップによる新しい独自設計型クラスターのシード](#)」を参照してください。キャッシュのデータをエクスポートするもう 1 つの理由は、.rdb ファイルをオフライン処理に使用することです。

Important

- ElastiCache バックアップとコピー先の Amazon S3 バケットは、同じ AWS リージョンに存在する必要があります。

Amazon S3 バケットにコピーされたバックアップは暗号化されますが、バックアップを保存する Amazon S3 バケットへのアクセス権を他の人に付与しないことを強くお勧めします。

- データ階層化を使用するクラスターでは、Amazon S3 へのバックアップのエクスポートはサポートされていません。詳細については、「[データ階層化](#)」を参照してください。
- バックアップのエクスポートは、Redis 独自設計型クラスター、Serverless Redis、および Serverless Memcached で使用できます。独自設計型 Memcached クラスターでは、バックアップをエクスポートできません。

Amazon S3 バケットにバックアップをエクスポートする前に、バックアップと同じ AWS リージョンに Amazon S3 バケットが必要です。バケット ElastiCache へのアクセスを許可します。最初の 2 つのステップで、これを行う方法を示します。

ステップ 1: Amazon S3 バケットを作成する

次の手順では、Amazon S3 コンソールを使用して、ElastiCache バックアップをエクスポートして保存する Amazon S3 バケットを作成します。

Amazon S3 バケットを作成するには

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. バケットの作成 を選択します。
3. バケットを作成する - バケット名と地域を選択する で、以下の操作を実行します。
 - a. [Bucket Name] に Amazon S3 バケットの名前を入力します。

Amazon S3 バケットの名前は DNS に準拠している必要があります。それ以外の場合は、バックアップファイルにアクセス ElastiCache できません。DNS コンプライアンスのルールは次のとおりです。

- 名前は、3～63 文字以内にする必要があります。
 - 名前は、ピリオド (.) で区切られた 1 つのラベルまたは一連の複数のラベルとして指定します。
 - 先頭の文字には小文字の英文字または数字を使用します。
 - 終了の文字には小文字の英文字または数字を使用します。
 - 小文字の英文字、数字、およびダッシュのみを含めます。
 - 名前は IP アドレスの形式にすることはできません (例: 192.0.2.0)。
- b. リージョンリストから、Amazon S3 バケットの AWS リージョンを選択します。この AWS リージョンは、エクスポートする ElastiCache バックアップと同じ AWS リージョンである必要があります。
 - c. 作成を選択します。

Amazon S3 バケットの作成の詳細については、Amazon Simple Storage Service ユーザーガイドの「[バケットの作成](#)」を参照してください。

ステップ 2: Amazon S3 バケット ElastiCache へのアクセスを許可する

ElastiCache が Amazon S3 バケットにスナップショットをコピーできるようにするには、バケットポリシーを更新してバケット ElastiCache へのアクセスを許可する必要があります。

Warning

Amazon S3 バケットにコピーされるバックアップは暗号化されますが、Amazon S3 バケットへのアクセス権を持つユーザーなら、誰でもデータにアクセスできます。したがって、こ

の Amazon S3 バケットへの不正アクセスを防ぐよう IAM ポリシーを設定することを強くお勧めします。詳細については、「Amazon S3 ユーザーガイド」の「[アクセス管理](#)」を参照してください。

Amazon S3 バケットで適切なアクセス許可を作成するには、以下の手順を実行します。

S3 バケット ElastiCache へのアクセスを許可するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. バックアップのコピー先とする Amazon S3 バケットの名前を選択します。これは、「[ステップ 1: Amazon S3 バケットを作成する](#)」で作成した S3 バケットとなります。
3. [Permissions] (許可) タブを選択し、[Permissions] (許可) で、[Access control list (ACL)] (アクセスコントロールリスト (ACL)) を選択して、[Edit] (編集) を選択します。
4. 次のオプションを使用して、被付与者の正規 ID
540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353 を追加します。
 - [Objects] (オブジェクト): [List] (リスト)、[Write] (書き込み)
 - [Bucket ACL] (バケット ACL): [Read] (読み取り) または [Write] (書き込み)

Note

- PDT GovCloud リージョンの場合、正規 ID は `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`。
- OSU GovCloud リージョンの場合、正規 ID は `c54286759d2a83da9c480405349819c993557275cf37d820d514b42da6893f5c`。

5. [保存] を選択します。

ステップ 3: ElastiCache バックアップをエクスポートする

これで、S3 バケットを作成し、そのバケットへのアクセス ElastiCache 許可を付与しました。次に、ElastiCache コンソール、AWS CLI、または ElastiCache API を使用してスナップショットをエクスポートできます。以下の例では、発信者の IAM アイデンティティに、次の S3 固有 IAM アクセス許可を追加で持っていることを前提としています。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketLocation",
      "s3:ListAllMyBuckets",
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObject",
      "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3::*:*"
  }]
}
```

オプトインリージョンの場合、S3 バケットの更新されたポリシーの例を以下に示します。(この例では、アジアパシフィック (香港) リージョンを使用しています。)

```
{
  "Version": "2012-10-17",
  "Id": "Policy15397346",
  "Statement": [
    {
      "Sid": "Stmt15399483",
      "Effect": "Allow",
      "Principal": {
        "Service": "elasticache.amazonaws.com"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::hkg-elasticache-backup",
        "arn:aws:s3:::hkg-elasticache-backup/*"
      ]
    },
    {
      "Sid": "Stmt15399484",
      "Effect": "Allow",
      "Principal": {
        "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
      },
      "Action": "s3:*",
```

```
    "Resource": [  
      "arn:aws:s3:::hkg-elasticache-backup",  
      "arn:aws:s3:::hkg-elasticache-backup/*"  
    ]  
  }  
]  
}
```

ElastiCache バックアップのエクスポート (コンソール)

次の手順では、コンソールを使用してバックアップ ElastiCache を Amazon S3 バケットにエクスポートし、の外部からアクセスできるようにします ElastiCache。Amazon S3 バケットは、ElastiCache バックアップと同じ AWS リージョンに存在する必要があります。

Amazon S3 バケットに ElastiCache バックアップをエクスポートするには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. バックアップのリストを表示するには、左のナビゲーションペインから [Backups] を選択します。
3. バックアップのリストから、エクスポートするバックアップの名前の左にあるチェックボックスをオンにします。
4. コピー を選択します。
5. バックアップのコピーを作成しますかダイアログボックスで、以下の設定を指定します。
 - a. [新しいバックアップ名] ボックスに、新しいバックアップの名前を入力します。

この名前は 1~1,000 文字で、UTF-8 エンコードが可能である必要があります。

ElastiCache は、ここに入力した値 .rdb にインスタンス識別子 と を追加します。たとえば、「my-exported-backup」と入力した場合、ElastiCache によって my-exported-backup-0001.rdb が作成されます。

- b. [Target S3 Location] リストから、バックアップをコピーする Amazon S3 バケット ([「ステップ 1: Amazon S3 バケットを作成する」](#) で作成したバケット) の名前を選択します。

エクスポートプロセスを成功させるには、ターゲット S3 ロケーションが、バックアップの AWS リージョンにある Amazon S3 バケットである必要があります。

- オブジェクトアクセス – 読み取り および 書き込み。

- アクセス許可 – 読み取り

詳細については、「[ステップ 2: Amazon S3 バケット ElastiCache へのアクセスを許可する](#)」を参照してください。

- c. コピー を選択します。

Note

S3 バケットにバックアップをエクスポート ElastiCache するために必要なアクセス許可がない場合、次のいずれかのエラーメッセージが表示されます。「[ステップ 2: Amazon S3 バケット ElastiCache へのアクセスを許可する](#)」に戻り、示されたアクセス権限を追加して、バックアップのエクスポートを再試行してください。

- ElastiCache は、S3 バケットに対する READ アクセス許可 %s を付与されていません。

解決策: バケットで読み取りのアクセス権限を追加します。

- ElastiCache は、S3 バケットに対する書き込みアクセス許可 %s を付与されていません。

解決策: バケットで書き込みのアクセス権限を追加します。

- ElastiCache は、S3 バケットで READ_ACP アクセス許可 %s を付与されていません。

解決策: バケットで読み取りのアクセス権限を追加します。

バックアップを別の AWS リージョンにコピーする場合は、Amazon S3 を使用してバックアップをコピーします。詳細については、Amazon Simple Storage Service ユーザーガイドの「[オブジェクトのコピー](#)」を参照してください。

ElastiCache サーバーレスバックアップのエクスポート (AWS CLI)

サーバーレスキャッシュのバックアップのエクスポート

以下のパラメータを指定して `export-serverless-cache-snapshot` CLI オペレーションを使用することで、Amazon S3 バケットにバックアップをエクスポートします。

パラメータ

- `--serverless-cache-snapshot-name` – コピーするバックアップの名前。

- `--s3-bucket-name` – バックアップをエクスポートする Amazon S3 バケットの名前。バックアップのコピーは、指定したバケットで作成されます。

エクスポートプロセスを成功させるには `--s3-bucket-name`、がバックアップの AWS リージョンにある Amazon S3 バケットである必要があります。

- オブジェクトアクセス – 読み取り および 書き込み。
- アクセス許可 – 読み取り

以下のオペレーションは、`my-s3-bucket` にバックアップをコピーします。

Linux、macOS、Unix の場合:

```
aws elasticache export-serverless-cache-snapshot \  
  --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 \  
  --s3-bucket-name my-s3-bucket
```

Windows の場合:

```
aws elasticache export-serverless-cache-snapshot ^  
  --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 ^  
  --s3-bucket-name my-s3-bucket
```

独自設計型 ElastiCache クラスターバックアップのエクスポート (AWS CLI)

独自設計型クラスターバックアップのエクスポート

以下のパラメータを指定して `copy-snapshot` CLI オペレーションを使用することで、Amazon S3 バケットにバックアップをエクスポートします。

パラメータ

- `--source-snapshot-name` – コピーするバックアップの名前。
- `--target-snapshot-name` – バックアップのコピーの名前。

この名前は 1~1,000 文字で、UTF-8 エンコードが可能である必要があります。

ElastiCache は、ここに入力した値 `.rdb` にインスタンス識別子 と を追加します。たとえば、「`my-exported-backup`」と入力した場合、ElastiCache によって `my-exported-backup-0001.rdb` が作成されます。

- `--target-bucket` – バックアップをエクスポートする Amazon S3 バケットの名前。バックアップのコピーは、指定したバケットで作成されます。

エクスポートプロセスを成功させるには `--target-bucket`、がバックアップの AWS リージョンにある Amazon S3 バケットである必要があります。

- オブジェクトアクセス – 読み取り および 書き込み。
- アクセス許可 – 読み取り

詳細については、「[ステップ 2: Amazon S3 バケット ElastiCache へのアクセスを許可する](#)」を参照してください。

以下のオペレーションは、`my-s3-bucket` にバックアップをコピーします。

Linux、macOS、Unix の場合:

```
aws elasticache copy-snapshot \  
  --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 \  
  --target-snapshot-name my-exported-backup \  
  --target-bucket my-s3-bucket
```

Windows の場合:

```
aws elasticache copy-snapshot ^  
  --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 ^  
  --target-snapshot-name my-exported-backup ^  
  --target-bucket my-s3-bucket
```

バックアップから新しいキャッシュへの復元

既存のバックアップを新しいサーバーレスキャッシュまたは独自設計型クラスターに復元できません。

サーバーレスキャッシュへのバックアップの復元 (コンソール)

Note

ElastiCache Serverless は、5.0 から利用可能な最新バージョンまでの Redis バージョンと互換性のある RDB ファイルをサポートしています。

サーバーレスキャッシュにバックアップを復元するには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで [バックアップ] を選択します。
3. バックアップのリストで、復元するバックアップ名の左にあるチェックボックスをオンにします。
4. [アクション] から [復元] を選択します。
5. 新しいサーバーレスキャッシュの名前と説明 (任意) を入力します。
6. [作成] をクリックして新しいキャッシュを作成し、バックアップからデータをインポートします。

独自設計型クラスターへのバックアップの復元 (コンソール)

独自設計型クラスターにバックアップを復元するには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで [バックアップ] を選択します。
3. バックアップのリストで、復元のバックアップ名の左にあるチェックボックスをオンにします。
4. [アクション] から [復元] を選択します。
5. [独自のキャッシュを設計] を選択し、ノードタイプ、サイズ、シャード数、レプリカ、AZ 配置、セキュリティ設定などのクラスター設定をカスタマイズします。

6. [作成] を選択して新しい独自設計型キャッシュを作成し、バックアップからデータをインポートします。

サーバーレスキャッシュへのバックアップの復元 (AWS CLI)

Note

ElastiCache Serverless は、5.0 から利用可能な最新バージョンまでの Redis バージョンと互換性のある RDB ファイルをサポートしています。

サーバーレスキャッシュにバックアップを復元するには (AWS CLI)

次の AWS CLI 例では、を使用して新しいキャッシュを作成し create-serverless-cache、バックアップからデータをインポートします。

Linux、macOS、Unix の場合:

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name CacheName \  
  --engine redis \  
  --snapshot-arns-to-restore Snapshot-ARN
```

Windows の場合:

```
aws elasticache create-serverless-cache ^ \  
  --serverless-cache-name CacheName ^ \  
  --engine redis ^ \  
  --snapshot-arns-to-restore Snapshot-ARN
```

Windows の場合:

独自設計型クラスターへのバックアップの復元 (AWS CLI)

独自設計型クラスターにバックアップを復元するには (AWS CLI)

Redis サーバーレスキャッシュバックアップを復元したり、Redis 独自設計型クラスターを再構築したりすることもできます。

Redis サーバーレスキャッシュバックアップは、2 つの方法で復元できます。

- AWS CLI オペレーション を使用して、単一ノード Redis (クラスターモードが無効) クラスターに復元できます create-cache-cluster。
- リードレプリカ (レプリケーショングループ) のある Redis クラスターに復元できます。これを行うには、AWS CLI オペレーションで Redis (クラスターモードが無効) または Redis (クラスターモードが有効) のいずれかを使用できます create-replication-group。この場合、Redis .rdb ファイルを使用して復元をシードします。新しい独自設計型クラスターをシードする方法の詳細については、「[外部で作成されたバックアップによる新しい独自設計型クラスターのシード](#)」を参照してください。

Redis (クラスターモードが無効) バックアップは 2 つの方法で復元できます。

- AWS CLI オペレーション を使用して、単一ノード Redis (クラスターモードが無効) クラスターに復元できます create-cache-cluster。
- リードレプリカ (レプリケーショングループ) のある Redis クラスターに復元できます。これを行うには、AWS CLI オペレーションで Redis (クラスターモードが無効) または Redis (クラスターモードが有効) のいずれかを使用できます create-replication-group。この場合、Redis .rdb ファイルを使用して復元をシードします。新しい独自設計型クラスターをシードする方法の詳細については、「[外部で作成されたバックアップによる新しい独自設計型クラスターのシード](#)」を参照してください。

create-cache-cluster または create-replication-group オペレーションを使用する場合、必ずパラメータ --snapshot-name または --snapshot-arn を含めて、新しいクラスターまたはレプリケーショングループにバックアップからのデータをシードします。

バックアップの削除

自動バックアップは、保持期限を過ぎると自動的に削除されます。クラスターを削除すると、そのクラスターのすべての自動バックアップも削除されます。レプリケーショングループを削除すると、そのグループのクラスターからすべて自動バックアップも削除されます。

ElastiCache には、バックアップが自動作成されたか手動で作成されたかにかかわらず、いつでもバックアップを削除できる削除 API オペレーションが用意されています。(手動バックアップには保持期限がないため、手動削除は手動スナップショットを削除する唯一の方法です)。

コンソール、ElastiCache、または ElastiCache API を使用して AWS CLI バックアップを削除できます。

バックアップの削除 (コンソール)

次の手順では、コンソールを使用してバックアップを削除します ElastiCache。

バックアップを削除するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[バックアップ] を選択します。
[バックアップ] 画面にバックアップのリストが表示されます。
3. 削除するバックアップの名前の左にあるチェックボックスをオンにします。
4. [削除] をクリックします。
5. このバックアップを削除する場合は、[バックアップの削除] 確認画面で [削除] を選択します。ステータスが deleting に変わります。

サーバーレスバックアップの削除 (AWS CLI)

サーバーレスバックアップを削除するには、次のパラメータを指定して delete-snapshot AWS CLI オペレーションを使用します。

- --serverless-cache-snapshot-name – 削除するバックアップの名前。

以下のコードはバックアップ myBackup を削除します。

```
aws elasticache delete-serverless-cache-snapshot --serverless-cache-snapshot-name myBackup
```

詳細については、AWS CLI コマンドリファレンスの「[delete-serverless-cache-snapshot](#)」を参照してください。

独自設計型クラスターバックアップの削除 (AWS CLI)

以下のパラメータを指定して delete-snapshot AWS CLI オペレーションを使用して、独自設計型クラスターバックアップを削除します。

- --snapshot-name – 削除するバックアップの名前。

以下のコードはバックアップ myBackup を削除します。

```
aws elasticache delete-snapshot --snapshot-name myBackup
```

詳細については、AWS CLI CLI コマンドリファレンスの「[delete-snapshot](#)」を参照してください。

バックアップへのタグ付け

タグ形式で各バックアップに独自のメタデータを割り当てることができます。タグを使用すると、バックアップを用途、所有者、環境などのさまざまな方法で分類できます。これは、同じタイプのリソースが多数ある場合に役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。詳細については、「[タグを付けることができるリソース](#)」を参照してください。

コスト配分タグは、コストをタグ値別に請求書にグループ化することで、複数の AWS のサービスにわたるコストを追跡する手段です。コスト配分タグの詳細については、「[コスト配分タグの使用](#)」を参照してください。

ElastiCache コンソール、または ElastiCache API を使用して AWS CLI、バックアップのコスト配分タグを追加、一覧表示、変更、削除、またはコピーできます。詳細については、「[コスト配分タグによるコストのモニタリング](#)」を参照してください。

外部で作成されたバックアップによる新しい独自設計型クラスタのシード

新しい Redis 独自設計型クラスタを作成するときに、Redis .rdb バックアップファイルのデータでシードできます。クラスタのシードは、現在の外部で Redis インスタンスを管理 ElastiCache していて、Redis の独自設計型クラスタ ElastiCache に既存の Redis データを入力する場合に便利です。

Amazon 内で作成された Redis バックアップから新しい Redis 独自設計型クラスタをシードするには ElastiCache、[「」](#)を参照してください[バックアップから新しいキャッシュへの復元](#)。

新しい Redis .rdb ファイルを使用して新しい Redis 独自設計型クラスタをシードするときは、以下を実行できます。

- パーティション分割されていないクラスタから、Redis バージョン 3.2.4 を実行している Redis (クラスタモードが有効) 独自設計型クラスタにアップグレードする。
- 新しい独自設計型クラスタのシャード (API および CLI ではノードグループと呼ばれる) の数を指定する。この数は、バックアップファイルの作成に使用された独自設計型クラスタ内のシャードの数とは異なる場合があります。
- 新しい独自設計型クラスタに異なるノードタイプを指定する。つまり、バックアップを作成したクラスタで使用されているノードタイプよりも大きい小さいかを指定します。より小さいノードタイプにスケールダウンする場合は、新しいノードタイプに、データと Redis のオーバーヘッドに対する十分なメモリがあることを確認してください。詳細については、[「Redis スナップショットを作成するのに十分なメモリがあることの確認」](#)を参照してください。
- バックアップファイルの作成に使用されたクラスタとは異なる新しい Redis (クラスタモードが有効) クラスタのスポットで、キーを分散します。

Note

Redis (クラスタモードが無効) クラスタから作成された .rdb ファイルから Redis (クラスタモードが無効) クラスタをシードすることはできません。

⚠ Important

- Redis バックアップデータがノードのリソースを超えていないことを確認する必要があります。たとえば、2.9 GB のメモリがある cache.m3.medium ノードに、5 GB の Redis データがある .rdb ファイルをアップロードすることはできません。

バックアップが大きすぎる場合、クラスターのステータスは `restore-failed` になります。その場合は、クラスターを削除してやり直す必要があります。

ノードタイプと仕様の完全なリストについては、[Redis のノードタイプ固有のパラメータ](#)「」および「[Amazon ElastiCache 製品の機能と詳細](#)」を参照してください。

- Redis .rdb ファイルは、Amazon S3 サーバー側の暗号化 (SSE-S3) でのみ暗号化できます。詳細については、「[サーバー側の暗号化を使用したデータの保護](#)」を参照してください。

以下は、Redis クラスターを Redis ElastiCache の外部から for Redis に移行する手順 ElastiCache を説明するトピックです。

ElastiCache for Redis への移行

- [ステップ 1: Redis バックアップを作成する](#)
- [ステップ 2: Amazon S3 バケットとフォルダを作成する](#)
- [ステップ 3: バックアップを Amazon S3 にアップロードする](#)
- [ステップ 4: .rdb ファイルへの ElastiCache 読み取りアクセスを許可する](#)

ステップ 1: Redis バックアップを作成する

ElastiCache for Redis インスタンスをシードする Redis バックアップを作成するには

1. 既存の Redis インスタンスに接続します。
2. Redis BGSAVE オペレーションまたは SAVE オペレーションを実行してバックアップを作成します。 .rdb ファイルの場所を書き留めておきます。

BGSAVE は非同期処理であり、処理中も他のクライアントをブロックしません。詳細については、Redis ウェブサイトの「[BGSAVE](#)」を参照してください。

SAVE が同期され、完了するまで他のプロセスがブロックされます。詳細については、Redis ウェブサイトの「[保存](#)」を参照してください。

バックアップの作成の詳細については、Redis ウェブサイトの「[Redis Persistence](#)」を参照してください。

ステップ 2: Amazon S3 バケットとフォルダを作成する

バックアップファイルを作成したら、Amazon S3 バケット内のフォルダにアップロードする必要があります。これを行うには、最初にそのバケット内に Amazon S3 バケットとフォルダが必要です。既に適切なアクセス許可を持つ Amazon S3 バケットフォルダがある場合は、「[ステップ 3: バックアップを Amazon S3 にアップロードする](#)」に進むことができます。

Amazon S3 バケットを作成するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. Amazon S3 バケットを作成するには、Amazon Simple Storage Service ユーザーガイドの「[バケットの作成](#)」の手順に従います。

Amazon S3 バケットの名前は DNS に準拠している必要があります。それ以外の場合は、バックアップファイルにアクセス ElastiCache できません。DNS コンプライアンスのルールは次のとおりです。

- 名前は、3~63 文字以内にする必要があります。
- 名前は、ピリオド (.) で区切られた 1 つのラベルまたは一連の複数のラベルとして指定します。
 - 先頭の文字には小文字の英文字または数字を使用します。
 - 終了の文字には小文字の英文字または数字を使用します。
 - 小文字の英文字、数字、およびダッシュのみを含めます。
- 名前は IP アドレスの形式にすることはできません (例: 192.0.2.0)。

Amazon S3 バケットは、Redis クラスター ElastiCache 用の新しいと同じ AWS リージョンに作成する必要があります。このアプローチにより、が Amazon S3 から .rdb ファイルを ElastiCache 読み取りたときのデータ転送速度が最高になります。

Note

データを可能な限り安全に保つには、Amazon S3 バケットに対するアクセス許可をできるだけ制限します。同時に、バケットとその内容を使用して新しい Redis クラスターをシードするためのアクセス許可を付与する必要があります。

Amazon S3 バケットにフォルダを追加するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. .rdb ファイルのアップロード先となるバケットの名前を選択します。
3. Create folder (フォルダの作成) を選択します。
4. 新しいフォルダの名前を入力します。
5. 保存 を選択します。

バケット名とフォルダ名の両方の名前を書き留めます。

ステップ 3: バックアップを Amazon S3 にアップロードする

次に、「[ステップ 1: Redis バックアップを作成する](#)」で作成した .rdb ファイルをアップロードします。アップロード先は、「[ステップ 2: Amazon S3 バケットとフォルダを作成する](#)」で作成した Amazon S3 バケットとフォルダです。このタスクの詳細については、「[バケットへのオブジェクトの追加](#)」を参照してください。ステップ 2 と 3 の間に、作成したフォルダ名を選択します。

.rdb ファイルを Amazon S3 フォルダにアップロードするには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. ステップ 2 で作成した Amazon S3 バケットの名前を選択します。
3. ステップ 2 で作成したフォルダの名前を選択します。
4. アップロードを選択します。
5. ファイルの追加を選択します。
6. アップロードする 1 つまたは複数のファイルを参照して見つけ、そのファイルを選択します。複数のファイルを選択するには、Ctrl キーを押しながら各ファイル名を選択します。

7. **開く** をクリックします。
8. 正しいファイルが [アップロード] ダイアログボックスに表示されることを確認してから、[アップロード] を選択します。

.rdb ファイルへのパスを記録します。たとえば、バケット名が myBucket で、パスが myFolder/redis.rdb の場合は、「myBucket/myFolder/redis.rdb」と入力します。新しいクラスターにこのバックアップのデータをシードする際にこのパスが必要です。

詳細については、Amazon Simple Storage Service ユーザーガイドの [Bucket restrictions and limitations](#) を参照してください。

ステップ 4: .rdb ファイルへの ElastiCache 読み取りアクセスを許可する

次に、.rdb バックアップファイルへの ElastiCache 読み取りアクセス権を付与します。バックアップファイル ElastiCache へのアクセスは、バケットがデフォルト AWS リージョンにあるかオプトイン AWS リージョンにあるかに応じて、別の方法で許可します。

AWS 2019 年 3 月 20 日より前に導入されたリージョンは、デフォルトで有効になっています。これらの AWS リージョンですぐに作業を開始できます。2019 年 3 月 20 日以降に導入されたアジアパシフィック (香港) および中東 (バーレーン) などのリージョンは、デフォルトで無効になっています。AWS 全般のリファレンスの「[AWS リージョンの管理](#)」で説明されているように、これらのリージョンを使用する前に、それらを有効にするか、オプトインする必要があります。

AWS リージョンに応じてアプローチを選択します。

- デフォルトリージョンの場合は、「[デフォルトリージョンの .rdb ファイルへの ElastiCache 読み取りアクセスを許可する](#)」の手順を使用します。
- オプトインリージョンの場合は、「[オプトインリージョンの .rdb ファイルへの ElastiCache 読み取りアクセスを許可する](#)」の手順を使用します。

デフォルトリージョンの .rdb ファイルへの ElastiCache 読み取りアクセスを許可する


AWS 2019 年 3 月 20 日より前に導入されたリージョンは、デフォルトで有効になっています。これらの AWS リージョンですぐに作業を開始できます。2019 年 3 月 20 日以降に導入されたアジアパシフィック (香港) および中東 (バーレーン) などのリージョンは、デフォルトで無効になっています。AWS 全般のリファレンスの「[AWS リージョンの管理](#)」で説明されているように、これらのリージョンを使用する前に、それらを有効にするか、オプトインする必要があります。

デフォルトで有効になっている AWS リージョンのバックアップファイルへの ElastiCache 読み取りアクセスを許可するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
2. .rdb ファイルを含む S3 バケットの名前を選択します。
3. .rdb ファイルを含むフォルダの名前を選択します。
4. .rdb バックアップファイルの名前を選択します。選択したファイルの名前は、ページ先頭のタブの上に表示されます。
5. [アクセス許可] を選択します。
6. aws-scs-s3-readonly または次のリストの正規化 ID の 1 つがユーザーとして表示されていない場合は、以下の作業を行います。
 - a. 他の AWS アカウントのアクセスで、被付与者の追加を選択します。
 - b. ボックスに、次に示すように AWS リージョンの正規 ID を追加します。

- AWS GovCloud (米国西部) リージョン :

```
40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6
```

 Important

バックアップを の Redis クラスターにダウンロード AWS GovCloud (US) するには、バックアップを の S3 バケットに配置する必要があります AWS GovCloud (US)。

- AWS デフォルトで有効になっているリージョン :

```
540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353
```

- c. 以下について、[はい] を選択してバケットのアクセス許可を設定します。
 - [List/write object] (オブジェクトのリスト化/書き込み)
 - [Read/write object ACL permissions] (オブジェクト ACL の読み取り/書き込みアクセス許可)
 - d. [保存] を選択します。
7. [概要] を選択し、[ダウンロード] を選択します。

オプトインリージョンの .rdb ファイルへの ElastiCache 読み取りアクセスを許可する

AWS 2019 年 3 月 20 日より前に導入されたリージョンは、デフォルトで有効になっています。これらの AWS リージョンですぐに作業を開始できます。2019 年 3 月 20 日以降に導入されたアジアパシフィック (香港) および中東 (バーレーン) などのリージョンは、デフォルトで無効になっています。AWS 全般のリファレンスの「[AWS リージョンの管理](#)」で説明されているように、これらのリージョンを使用する前に、それらを有効にするか、オプトインする必要があります。

次に、.rdb バックアップファイルへの ElastiCache 読み取りアクセス権を付与します。

バックアップファイルへの ElastiCache 読み取りアクセスを許可するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
 2. .rdb ファイルを含む S3 バケットの名前を選択します。
 3. .rdb ファイルを含むフォルダの名前を選択します。
 4. .rdb バックアップファイルの名前を選択します。選択したファイルの名前は、ページ先頭のタブの上に表示されます。
 5. アクセス許可 タブを選択します。
 6. 許可 で、バケットポリシーを選択し、編集を選択します。
 7. ポリシーを更新して、オペレーションを実行する ElastiCache ために必要なアクセス許可を付与します。
- ["Service" : "*region-full-name*.elasticache-snapshot.amazonaws.com"] を Principal に追加します。
 - スナップショットを Amazon S3 バケットにエクスポートするために必要な、以下のアクセス許可を追加します。
 - "s3:GetObject"
 - "s3:ListBucket"
 - "s3:GetBucketAcl"

次に、更新されたポリシーの例を示します。

```
{
  "Version": "2012-10-17",
  "Id": "Policy15397346",
```



```
"Statement": [
  {
    "Sid": "Stmt15399483",
    "Effect": "Allow",
    "Principal": {
      "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
    },
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketAcl"
    ],
    "Resource": [
      "arn:aws:s3:::example-bucket",
      "arn:aws:s3:::example-bucket/backup1.rdb",
      "arn:aws:s3:::example-bucket/backup2.rdb"
    ]
  }
]
```

8. [変更の保存] をクリックします。

ステップ 5: .rdb ファイルデータを使用して ElastiCache クラスターをシードする

これで、ElastiCache クラスターを作成し、.rdb ファイルからのデータでシードする準備が整いました。クラスターを作成するには、「[クラスターの作成](#)」または「[ゼロからの Redis レプリケーショングループの作成](#)」の手順に従います。Redis がクラスターエンジンとして選択されていることを確認してください。

Amazon S3 にアップロードした Redis バックアップ ElastiCache の場所を特定するために使用する方法は、クラスターの作成に使用する方法によって異なります。

for ElastiCache Redis クラスターまたはレプリケーショングループを .rdb ファイルデータでシードしました

• ElastiCache コンソールの使用

[Cluster settings] (クラスター設定) を選択するときは、クラスターの作成方法として [Restore from backups] (バックアップから復元) を選択し、次に [Backup source] (バックアップソース) セクションで [Source] (ソース) として [Other backups] (その他のバックアップ) を選択します。[シード RDB ファイルの S3 ロケーション] ボックスに、ファイルの Amazon S3 パスを入力します。複数

の .rdb ファイルがある場合は、カンマ区切りのリストで各ファイルのパスを入力します。Amazon S3 パスは `myBucket/myFolder/myBackupFilename.rdb` のようになります。

- の使用 AWS CLI

`create-cache-cluster` または `create-replication-group` オペレーションを使用する場合、パラメータ `--snapshot-arns` を使用して、各 .rdb ファイルの完全修飾 ARN を指定します。例えば、`arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`。ARN は、Amazon S3 に保存したバックアップファイルに解決される必要があります。

- ElastiCache API の使用

`CreateCacheCluster` または `CreateReplicationGroup` ElastiCache API オペレーションを使用する場合は、パラメータを使用して `SnapshotArns`、各 .rdb ファイルに対して完全修飾 ARN を指定します。例えば、`arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`。ARN は、Amazon S3 に保存したバックアップファイルに解決される必要があります。

Important

Redis (クラスターモードが有効) クラスターを提携させる場合は、新しいクラスターまたはレプリケーショングループの各ノードグループ (シャード) を設定する必要があります。これを行うには、パラメータ `--node-group-configuration` (API: `NodeGroupConfiguration`) を使用します。詳細については、次を参照してください。

- CLI: AWS CLI リファレンスの [create-replication-group](#)
- API: ElastiCache API リファレンスの [CreateReplicationグループ](#)

クラスターの作成処理中、Redis バックアップ内のデータがクラスターに書き込まれます。ElastiCache イベントメッセージを表示することで、進行状況をモニタリングできます。これを行うには、ElastiCache コンソールを参照し、**キャッシュイベント** を選択します。AWS ElastiCache コマンドラインインターフェイスまたは ElastiCache API を使用してイベントメッセージを取得することもできます。詳細については、「[ElastiCache イベントの表示](#)」を参照してください。

エンジンバージョンとアップグレード

このセクションでは、サポートされる Redis エンジンのバージョンとアップグレードする方法について説明します。

トピック

- [エンジンバージョンとアップグレード](#)
- [サポートされている ElastiCache for Redis のバージョン](#)
- [Redis バージョンのサポート終了スケジュール](#)
- [エンジンバージョンのアップグレード方法](#)
- [ブロックされた Redis エンジンのアップグレードの解決](#)
- [メジャーバージョンの動作と互換性の違い](#)

エンジンバージョンとアップグレード

ElastiCache for Redis のバージョンは、メジャーとマイナーコンポーネントで構成されるセマンティックバージョンで識別されます。例えば、Redis 6.2 では、メジャーバージョンは 6、マイナーバージョンは 2 です。独自設計型クラスターを操作する場合、ElastiCache for Redis は Redis 6.2.1 などの PATCH コンポーネントも公開します。パッチバージョンは 1 です。

メジャーバージョンは API 非互換の変更用であり、マイナーバージョンは下位互換性のある方法で追加された新機能用です。パッチバージョンは、下位互換性のあるバグ修正と機能以外の変更用です。

ElastiCache サーバーレスのバージョン管理

ElastiCache サーバーレスは、アプリケーションに影響を与えたり、ダウンタイムを発生させたりすることなく、最新のマイナーおよびパッチソフトウェアバージョンをキャッシュに自動的に適用します。ユーザー操作は必要はありません。

新しいメジャーバージョンがリリースされると、ElastiCache サーバーレスからコンソールに通知が送信され、イベントが EventBridge に送信されます。コンソール、CLI、または API を使用してキャッシュを変更し、最新のエンジンバージョンを選択することで、キャッシュを最新のメジャーバージョンにアップグレードできます。

独自設計型 ElastiCache クラスターのバージョン管理

独自設計型 ElastiCache クラスターで作業する場合、キャッシュクラスターを動かすソフトウェアを、ElastiCache でサポートされる新しいバージョンにアップグレードするタイミングを制御できます。キャッシュを最新のメジャーバージョン、マイナーバージョン、パッチバージョンにアップグレードするタイミングを制御できます。クラスターまたはレプリケーショングループを変更し、新しいエンジンのバージョンを指定することで、クラスターまたはレプリケーショングループに対するエンジンのバージョンのアップグレードを開始します。

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

バージョンのアップグレードは互換性のリスクがあるため、自動では実行されません。それらを自分で開始する必要があります。

クラスターまたはレプリケーショングループを変更し、新しいエンジンのバージョンを指定することで、クラスターまたはレプリケーショングループに対するエンジンのバージョンのアップグレードを開始します。詳細については、次を参照してください:

- [クラスターの変更](#)
- [レプリケーショングループの変更](#)

独自設計型クラスターを使用する際のアップグレードに関する考慮事項

Note

以下の考慮事項は、独自設計型クラスターをアップグレードする場合にのみ適用されません。ElastiCache サーバーレスには適用されません。

独自設計型クラスターをアップグレードする際は、以下を考慮してください。

- エンジンのバージョンアップは、パッチの適用方法をできる限り制御できるように設計されています。ただし、ElastiCache には、システムまたはキャッシュソフトウェアに重大なセキュリティ脆弱性が発生する可能性が低い場合に、お客様に代わってクラスターにパッチを適用するための権限があります。
- ElastiCache for Redis では、Redis 6.0 以降、複数のパッチバージョンを提供するのではなく、Redis OSS マイナーリリースごとに 1 つのバージョンが提供されます。
- Redis エンジンバージョン 5.0.6 以降では、最小限のダウンタイムでクラスターバージョンをアップグレードできます。このクラスターは、アップグレード中のすべての読み取りと、数秒かかるフェールオーバー操作中を除き、ほとんどすべてのアップグレード中の書き込みに対応します。
- 5.0.6 より前のバージョンで ElastiCache クラスターをアップグレードすることもできます。関連するプロセスは同じですが、DNS の伝播中にフェールオーバー時間が長くなる可能性があります (30 秒 ~ 1 分)。
- Redis 7 から、ElastiCache for Redis は、Redis (クラスターモードが無効) と Redis (クラスターモードが有効) の切り替えをサポートしています。
- Amazon ElastiCache for Redis エンジンのアップグレードプロセスは、既存のデータをベストエフォートで保持するように設計されており、Redis レプリケーションが正常に実行される必要があります。
- エンジンをアップグレードすると、ElastiCache for Redis は既存のクライアント接続を終了します。エンジンのアップグレード中のダウンタイムを最小限に抑えるため、エラー再試行とエクスポ

ネンシャルバックオフによる[Redis クライアントのベストプラクティス](#)と、[メンテナンス中のダウンタイムを最小限に抑える](#)ためのベストプラクティスを実装することをお勧めします。

- エンジンをアップグレードすると、Redis (クラスターモード無効) から Redis (クラスターモード有効) に直接アップグレードすることはできません。以下の手順では、Redis (クラスターモードが無効) から Redis (クラスターモードが有効) にアップグレードする方法を示しています。

Redis (クラスターモード無効) から Redis (クラスターモードが有効) エンジンバージョンにアップグレードするには

1. Redis (クラスターモードが無効) クラスターまたはレプリケーショングループのバックアップを作成します。詳細については、「[手動バックアップの取得](#)」を参照してください。
 2. バックアップを使用して、1つのシャード(ノードグループ)を持つ Redis (クラスターモードが有効) クラスターを作成してシードします。新しいエンジンのバージョンを指定し、クラスターまたはレプリケーショングループの作成時にクラスターモードを有効にします。詳細については、「[外部で作成されたバックアップによる新しい独自設計型クラスターのシード](#)」を参照してください。
 3. 古い Redis (クラスターモードが無効) クラスターまたはレプリケーショングループを削除します。詳細については、「[クラスターの削除](#)」または「[レプリケーショングループの削除](#)」を参照してください。
 4. 新しい Redis (クラスターモードが有効) クラスターまたはレプリケーショングループを、必要なシャード(ノードグループ)の数に合わせてスケールします。詳細については、[Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#) を参照してください。
- 例えば、5.0.6 から 6.0 にメジャーエンジンのバージョンをアップグレードする場合は、新しいエンジンバージョンと互換性のある新しいパラメータグループも選択する必要があります。
 - 単一の Redis クラスターや、マルチ AZ が無効になっているクラスターの場合、「[Redis スナップショットを作成するのに十分なメモリがあることの確認](#)」で説明されているように、Redis 用に十分なメモリを確保することをお勧めします。このような場合、プライマリはアップグレードプロセスの実行中、リクエストに対応できません。
 - マルチ AZ が有効になっている Redis クラスターの場合、書き込みの受信トラフィックが少ない期間中にエンジンのアップグレードを予定することもお勧めします。Redis 5.0.6 以降にアップグレードする場合は、アップグレードプロセス中も、プライマリクラスターで引き続きサービスリクエストを利用できます。

複数のシャードを含むクラスターおよびレプリケーショングループは、次のように処理および修正されます。

- すべてのシャードは並行して処理されます。シャードでは、いつでも 1 つのアップグレードオペレーションのみが実行されます。
- 各シャードでは、プライマリが処理される前にすべてのレプリカが処理されます。シャードにレプリカが少ない場合、他のシャードのレプリカが処理を終了する前に、そのシャードのプライマリが処理されることがあります。
- すべてのシャード間で、プライマリノードはシリーズで処理されます。一度にアップグレードできるプライマリノードは 1 つだけです。
- 現在のクラスターまたはレプリケーショングループで暗号化が有効になっている場合、暗号化をサポートしていないエンジンバージョン (3.2.6 から 3.2.10 など) にアップグレードすることはできません。

エンジンバージョンのアップグレード方法

クラスターまたはレプリケーショングループのバージョンアップは、ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、より新しいエンジンバージョンを指定することで開始します。詳細については、以下のトピックを参照してください。

クラスターおよびレプリケーショングループを変更する方法

クラスター	レプリケーショングループ
の使用 AWS Management Console	の使用 AWS Management Console
の使用 AWS CLI	の使用 AWS CLI
ElastiCache API の使用	ElastiCache API の使用

ブロックされた Redis エンジンのアップグレードの解決

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

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

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

ブロックされた Redis エンジンのアップグレードを解決するには

- 次のいずれかを行います:

- [すぐに適用] チェックボックスをオフにすることで、Redis エンジンのアップグレードオペレーションを次のメンテナンスウィンドウに予定します。

CLI では、`--no-apply-immediately` を使用します。API では、`ApplyImmediately=false` を使用します。

- Redis のエンジンアップグレードオペレーションを実行する次のメンテナンス期間 (またはその後) まで待ちます。
- [すぐに適用] チェックボックスをオンにすることで、Redis のスケールアップオペレーションをこのクラスターの変更に追加します。

CLI では、`--apply-immediately` を使用します。API では、`ApplyImmediately=true` を使用します。

このアプローチにより、エンジンのアップグレードがすぐに実行されて、次のメンテナンスウィンドウ中のエンジンのアップグレードはキャンセルされます。

サポートされている ElastiCache for Redis のバージョン

ElastiCache サーバーレスキャッシュは、次の Redis バージョンをサポートしています。

- [ElastiCache for Redis バージョン 7.1 \(拡張\)](#)

独自設計型 ElastiCache クラスターは、次の Redis バージョンをサポートします。

- [ElastiCache for Redis バージョン 7.1 \(拡張\)](#)
- [ElastiCache for Redis バージョン 7.0 \(拡張\)](#)
- [ElastiCache for Redis バージョン 6.2 \(拡張\)](#)
- [ElastiCache for Redis バージョン 6.0 \(拡張\)](#)
- [ElastiCache for Redis バージョン 5.0.6 \(拡張\)](#)
- [ElastiCache for Redis バージョン 5.0.5 \(廃止、バージョン 5.0.6 を使用してください\)](#)
- [ElastiCache for Redis バージョン 5.0.4 \(廃止、バージョン 5.0.6 を使用してください\)](#)
- [ElastiCache for Redis バージョン 5.0.3 \(廃止、バージョン 5.0.6 を使用してください\)](#)
- [ElastiCache for Redis バージョン 5.0.0 \(廃止、バージョン 5.0.6 を使用してください\)](#)
- [ElastiCache for Redis バージョン 4.0.10 \(拡張\)](#)
- [過去のサポート終了 \(EOL\) バージョン \(3.x\)](#)
- [過去のサポート終了 \(EOL\) バージョン \(2.x\)](#)

ElastiCache for Redis バージョン 7.1 (拡張)

このリリースには、ワークロードのスループットを向上させ、操作のレイテンシーを低減するためのパフォーマンスの改善が含まれています。ElastiCache 7.1 では、[主に次の 2 つの機能強化](#)が導入されています。

拡張された I/O スレッド機能を拡張して、プレゼンテーション層のロジックも処理できるようにしました。プレゼンテーション層とは、クライアント入力を読み取るだけでなく、入力を Redis バイナリコマンド形式に解析する拡張 I/O スレッドを指します。その後、これがメインスレッドに転送されて実行され、パフォーマンスが向上します。Redis のメモリアクセスパターンが改善されました。多くのデータ構造操作の実行ステップをインターリーブすることで、並列メモリアクセスとメモリアクセスのレイテンシーの短縮を実現しています。Graviton3 ベースの R7g.4xlarge 以上で ElastiCache を実行する場合、お客様はノードあたり毎秒 100 万件を超えるリクエストを処理でき

まず、ElastiCache for Redis v7.1 のパフォーマンスが向上したことで、ElastiCache for Redis v7.0 と比較してスループットが最大 100% 向上し、P99 のレイテンシーを 50% 削減できます。これらの機能強化は、CPU タイプに関係なく、物理コアが 8 個以上 (Graviton では 2xlarge、x86 では 4xlarge) のノードサイズで有効になり、クライアントを変更する必要がありません。

Note

ElastiCache v7.1 は OSS Redis v7.0 と互換性があります。

ElastiCache for Redis バージョン 7.0 (拡張)

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

- [Redis 関数](#): ElastiCache for Redis 7 では、Redis 関数のサポートが追加され、管理されたエクスペリエンスが提供されるため、開発者は ElastiCache クラスターに保存されたアプリケーションロジックを使用して [LUA スクリプト](#) を実行でき、クライアントは毎回の接続でスクリプトをサーバーに再送信する必要がありません。
- [ACL の改善](#): ElastiCache for Redis 7 では、次期バージョンの Redis アクセスコントロールリスト (ACL) のサポートが追加されました。ElastiCache for Redis 7 では、クライアントは Redis の特定のキーまたはキースペースに対して複数の権限セットを指定できるようになりました。
- [シャードされた Pub/Sub](#): ElastiCache for Redis 7 では、クラスターモード有効 (CME) で ElastiCache を実行する際に、Redis Pub/Sub 機能をシャードされた方法で実行するサポートが追加されました。Redis Pub/Sub 機能により、発行者はチャンネル上の任意の数のサブスクリバースにメッセージを発行できます。Amazon ElastiCache for Redis 7 では、チャンネルは ElastiCache クラスターのシャードにバインドされるため、シャード間でチャンネル情報を伝達する必要がなくなり、スケーラビリティが向上します。
- [拡張 I/O 多重化](#): ElastiCache for Redis バージョン 7 では、拡張された I/O 多重化が導入されています。これにより、ElastiCache クラスターへの多数の同時クライアント接続がある高スループットワークロードのスループットが向上し、レイテンシーが短縮されます。例えば、r6g.xlarge ノードのクラスターを使用し、5200 の同時クライアントを実行する場合、ElastiCache for Redis バージョン 6 と比較して、スループット (1 秒あたりの読み取りおよび書き込み操作) が最大 72% 向上し、P99 レイテンシーが最大 71% 減少します。

Redis 7.0 のリリースの詳細については、GitHub の Redis の「[Redis 7.0 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 6.2 (拡張)

ElastiCache for Redis 6.2 では、8 個以上の vCPU を持つ x86 ノードタイプ、または 4 個以上の vCPU を持つ Graviton2 ノードタイプを使用した TLS 対応クラスターのパフォーマンスが向上しています。これらの機能強化では、暗号化を他の vCPUs にオフロードすることでスループットが向上し、クライアントの接続確立時間が短縮されました。Redis 6.2 では、アクセスコントロールリスト (ACL) ルールを使用して Pub/Sub チャンネルへのアクセスを管理することもできます。

このバージョンでは、ローカルに接続された NVMe SSD を含む、クラスターノードでのデータ階層化のサポートも導入されています。詳細については、「[データ階層化](#)」を参照してください。

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

エンジンバージョンを指定するには、6.2 を使用します。ElastiCache for Redis は、利用可能な任意のパッチバージョンの Redis 6.2 を自動的に呼び出します。例えば、キャッシュクラスターを作成または変更すると、`--engine-version` パラメータは 6.2 に設定されます。クラスターは、作成時および変更時に、現在利用可能な任意のパッチバージョンの Redis 6.2 で起動されます。API でエンジンバージョン 6.x を指定すると、最新のマイナーバージョンの Redis 6 になります。

既存の 6.0 クラスターの場合

合、`CreateCacheCluster`、`ModifyCacheCluster`、`CreateReplicationGroup`、または `ModifyReplicationGroup` API で `AutoMinorVersionUpgrade` パラメータを `yes` に設定することで、次のマイナーバージョン自動アップグレードにオプトインできます。ElastiCache for Redis では、セルフサービスの更新を使用して、既存の 6.0 クラスターのマイナーバージョンを 6.2 にアップグレードします。詳細については、[Amazon ElastiCache でのセルフサービスの更新](#)をご覧ください。

`DescribeCacheEngineVersions` API の呼び出し時に、`EngineVersion` パラメータ値が 6.2 に設定され、パッチバージョンを含む実際のエンジンバージョンが `CacheEngineVersionDescription` フィールドに返されます。

Redis 6.2 のリリースの詳細については、GitHub の Redis の [Redis 6.2 リリースノート](#) を参照してください。

ElastiCache for Redis バージョン 6.0 (拡張)

Amazon ElastiCache for Redis には、Redis エンジンの次のバージョンが導入されています。これには、[ロールベースのアクセスコントロールによるユーザーの認証](#)、クライアント側のキャッシュ、および大幅な運用上の改善が含まれます。

ElastiCache for Redis では、Redis 6.0 以降、複数のパッチバージョンを提供するのではなく、Redis OSS マイナーリリースごとに 1 つのバージョンが提供されます。ElastiCache for Redis は、実行中のキャッシュクラスターのパッチバージョンを自動的に管理し、パフォーマンスの向上とセキュリティ強化を実現します。

また、AutoMinorVersionUpgrade パラメータを yes に設定して、次のマイナーバージョン自動アップグレードにオプトインすることもできます。ElastiCache for Redis は、セルフサービスの更新を通じてマイナーバージョンのアップグレードを管理します。詳細については、「[でのサービスの更新 ElastiCache](#)」を参照してください。

エンジンのバージョンを指定するには、6.0 を使用します。ElastiCache for Redis は、利用可能な任意のパッチバージョンの Redis 6.0 を自動的に呼び出します。例えば、キャッシュクラスターを作成または変更する場合は、`--engine-version` パラメータを 6.0 に設定します。クラスターは、作成時または変更時に、現在利用可能な Redis 6.0 の任意のパッチバージョンで起動されます。特定のパッチバージョン値を持つリクエストは拒否され、例外がスローされ、プロセスは失敗します。

DescribeCacheEngineVersions API の呼び出し時に、EngineVersion パラメータ値が 6.0 に設定され、パッチバージョンを含む実際のエンジンバージョンが CacheEngineVersionDescription フィールドに返されます。

Redis 6.0 のリリースの詳細については、GitHub の Redis の「[Redis 6.0 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 5.0.6 (拡張)

Amazon ElastiCache for Redis には、バグ修正および以下の累積更新を含む次のバージョンの Redis エンジンが導入されています。

- 特別な状況におけるエンジンの安定性の保証。
- Hyperloglog エラー処理の強化。
- 信頼性の高いレプリケーションを目的としたハンドシェイクコマンドの強化。
- XCLAIM コマンドを使用した一貫性のあるメッセージ配信の追跡。
- オブジェクトの LFU フィールド管理の強化。

- ZPOP 使用時のトランザクション管理の強化。
- コマンドの名前を変更する機能: `rename-commands` と呼ばれるパラメータを使用すると、`FLUSHALL` や `FLUSHDB` など、誤ってデータ損失を発生させることがある、危険または処理負荷が大きい可能性がある Redis コマンドの名前を変更できます。これは、オープンソース Redis の名前変更コマンドの設定と似ています。ただし、ElastiCache では、フルマネージド型のワークフローを提供することによりエクスペリエンスを向上させています。コマンド名の変更はすぐに適用され、クラスター内でコマンドリストを含むすべてのノードにわたって自動的に反映されます。ノードの再起動など、お客様による介入は必要ありません。

次の例では、既存のパラメータグループを変更する方法を示しています。これには `rename-commands` パラメータが含まれます。これは、名前を変更するコマンドのスペースで区切られたリストです。

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-name custom_param_group --parameter-name-values "ParameterName=rename-commands, ParameterValue='flushall restrictedflushall'" --region region
```

この例では、`rename-commands` パラメータは、`flushall` コマンドの名前を `restrictedflushall` に変更するために使用します。

複数のコマンドの名前を変更するには、以下を使用します。

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-name custom_param_group --parameter-name-values "ParameterName=rename-commands, ParameterValue='flushall restrictedflushall flushdb restrictedflushdb'" --region region
```

変更を元に戻すには、次に示すようにコマンドを再実行し、`ParameterValue` リストから、名前の変更を維持する値を除外します。

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-name custom_param_group --parameter-name-values "ParameterName=rename-commands, ParameterValue='flushall restrictedflushall'" --region region
```

この場合、`flushall` コマンドは `restrictedflushall` に名前が変更され、名前が変更されたその他のコマンドは元のコマンド名に戻されます。

Note

名前を変更する場合は、以下の制限があります。

- 名前を変更するすべてのコマンドは、英数字である必要があります。
- 新しいコマンド名の最大長は 20 文字の英数字です。
- コマンドの名前を変更する場合は、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必ず更新します。
- コマンドの使用を完全に防ぐには、以下に示すように、キーワード `blocked` を使用します。

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-name custom_param_group --parameter-name-values "ParameterName=rename-commands, ParameterValue='flushall blocked'" --region region
```

パラメータの変更の詳細、および名前変更の対象であるコマンドのリストについては、「[Redis 5.0.3 パラメータの変更](#)」を参照してください。

- Redis ストリーム: これによりモデル化されたログデータ構造では、プロデューサーが新しいアイテムをリアルタイムで追加できます。また、コンシューマーがブロッキング方式またはノンブロッキング方式でメッセージを使用できます。ストリームは、コンシューマーグループも許可します。コンシューマーグループは、メッセージの同じストリームのさまざまな部分を共同で使用するクライアントのグループを表しています ([Apache Kafka](#) と同様)。詳細については、「[Redis ストリームの紹介](#)」を参照してください。
- XADD、XRANGE、XREAD など、ストリームコマンドファミリーのサポート。詳細については、「[Redis ストリームコマンド](#)」を参照してください。
- 多数の新しいパラメータと名前変更されたパラメータ。詳細については、「[Redis 5.0.0 パラメータの変更](#)」を参照してください。
- 新しい Redis メトリクス `StreamBasedCmds`。
- Redis ノードのスナップショット時間がわずかに短縮。

⚠ Important

Amazon ElastiCache for Redis は、[Redis オープンソースバージョン 5.0.1](#) から 2 つの重要なバグ修正をバックポートしました。以下に示します。

- 特定のキーの有効期限がすでに切れている場合、RESTORE が返信に一致しません。
- XCLAIM コマンドは、間違っただけのエントリを返したり、プロトコルを同期解除したりすることがあります。

これらのバグ修正はどちらも Redis エンジンバージョン 5.0.0 の ElastiCache for Redis サポートに含まれており、将来のバージョン更新で使用されます。

詳細については、GitHub の Redis の「[Redis 5.0.6 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 5.0.5 (廃止、バージョン 5.0.6 を使用してください)

Amazon ElastiCache for Redis には、次のバージョンの Redis エンジンが導入されています。これには、計画されたすべてのオペレーション中の自動フェイルオーバークラスターの ElastiCache for Redis のオンライン構成変更が含まれます。これで、クラスターをオンライン状態にしたまま、着信リクエストの処理を継続しながら、クラスターをスケーリングして Redis エンジンのバージョンをアップグレードし、パッチとメンテナンス更新を適用できるようになりました。また、バグ修正も含まれます。

詳細については、GitHub の Redis の「[Redis 5.0.5 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 5.0.4 (廃止、バージョン 5.0.6 を使用してください)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次のバージョンが導入されています。これには以下の機能強化が含まれています。

- 特別な状況におけるエンジンの安定性の保証。
- Hyperloglog エラー処理の強化。
- 信頼性の高いレプリケーションを目的としたハンドシェイクコマンドの強化。
- XCLAIM コマンドを使用した一貫性のあるメッセージ配信の追跡。
- オブジェクトの LFU フィールド管理の強化。
- ZPOP 使用時のトランザクション管理の強化。

詳細については、GitHub の Redis の「[Redis 5.0.4 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 5.0.3 (廃止、バージョン 5.0.6 を使用してください)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次のバージョンが導入されています。これにはバグ修正が含まれます。

ElastiCache for Redis バージョン 5.0.0 (廃止、バージョン 5.0.6 を使用してください)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次の主要バージョンが導入されています。ElastiCache for Redis 5.0.0 は次の強化機能をサポートします。

- Redis ストリーム: これによりモデル化されたログデータ構造では、プロデューサーが新しいアイテムをリアルタイムで追加できます。また、コンシューマーがブロッキング方式またはノンブロッキング方式でメッセージを使用できます。ストリームは、コンシューマーグループも許可します。コンシューマーグループは、メッセージの同じストリームのさまざまな部分を共同で使用するクライアントのグループを表しています ([Apache Kafka](#) と同様)。詳細については、「[Redis ストリームの紹介](#)」を参照してください。
- XADD、XRANGE、XREAD など、ストリームコマンドファミリーのサポート。詳細については、「[Redis ストリームコマンド](#)」を参照してください。
- 多数の新しいパラメータと名前変更されたパラメータ。詳細については、「[Redis 5.0.0 パラメータの変更](#)」を参照してください。
- 新しい Redis メトリクス StreamBasedCmds。
- Redis ノードのスナップショット時間がわずかに短縮。

ElastiCache for Redis バージョン 4.0.10 (拡張)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次の主要バージョンが導入されています。ElastiCache for Redis 4.0.10 は次の強化機能をサポートします。

- ElastiCache for Redis の単一バージョンでのオンラインクラスターのサイズ変更と暗号化。詳細については、次を参照してください:
 - [Redis \(クラスターモードが有効\) でのクラスターのスケーリング](#)
 - [Redis 用のオンラインリシャードイングおよびシャードの再分散 \(クラスターモードが有効\)](#)
 - [Amazon ElastiCache のデータセキュリティ](#)

- 多数の新しいパラメータ。詳細については、「[Redis 4.0.10 パラメータの変更](#)」を参照してください。
- MEMORY などのメモリコマンドファミリーのサポート。詳細については、[Redis コマンド](#) を参照してください (MEMO で検索)。
- オンライン中のメモリのデフラグメンテーションのサポート。これにより、メモリの使用が効率的になり、データに使用できるメモリが増えます。
- 非同期フラッシュと削除の Support。ElastiCache for Redis は、メインスレッドとは異なるスレッドで実行するために、UNLINK、FLUSHDB および FLUSHALL のようなコマンドをサポートしています。これにより、メモリが非同期的に解放されて、アプリケーションのパフォーマンスが上がり、レスポンス時間が短くなります。
- 新しい Redis メトリクス ActiveDefragHits。詳細については、「[Redis のメトリクス](#)」を参照してください。

Redis バージョン 3.2.10 を実行している Redis (クラスターモードが無効) ユーザーは、コンソールを使用して、オンラインアップグレードを介してクラスターをアップグレードできます。

ElastiCache for Redis クラスターのサイズ変更と暗号化サポートの比較

機能	3.2.6	3.2.10	4.0.10 以降
オンラインクラスターのサイズ変更 *	なし	はい	はい
転送時の暗号化 **	はい	いいえ	はい
保管時の暗号化 **	はい	いいえ	はい

* シャードを追加、削除、および負荷分散します。

** FedRAMP、HIPAA、および PCI DSS に準拠するアプリケーションで必要です。詳細については、「[Amazon のコンプライアンス検証 ElastiCache](#)」を参照してください。

過去のサポート終了 (EOL) バージョン (3.x)

ElastiCache for Redis バージョン 3.2.10 (拡張)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次の主要バージョンが導入されています。ElastiCache for Redis 3.2.10 にオンラインクラスターサ

サイズ変更が導入されました。これにより、クラスターが入力 I/O リクエストを処理しながら、クラスターからシャードを追加、または削除することができます。ElastiCache for Redis 3.2.10 のユーザーには、データを暗号化する機能を除き、Redis の以前のバージョンのすべての機能が提供されます。この機能は現在、バージョン 3.2.6 でのみ使用できます。

ElastiCache for Redis バージョン 3.2.6 と 3.2.10 の比較

機能	3.2.6	3.2.10
オンラインクラスターのサイズ変更 *	いいえ	はい
転送時の暗号化 **	はい	いいえ
保管時の暗号化 **	はい	いいえ

* シャードを追加、削除、および負荷分散します。

** FedRAMP、HIPAA、および PCI DSS に準拠するアプリケーションで必要です。詳細については、「[Amazon のコンプライアンス検証 ElastiCache](#)」を参照してください。

詳細については、次を参照してください:

- [Redis 用のオンラインリシャーディングおよびシャードの再分散 \(クラスターモードが有効\)](#)
- [オンラインクラスターのサイズ変更](#)

ElastiCache for Redis バージョン 3.2.6 (拡張)

Amazon ElastiCache for Redis では、Amazon ElastiCache でサポートされている Redis エンジンの次の主要バージョンが導入されています。ElastiCache for Redis 3.2.6 のユーザーには、以前のバージョンの Redis のすべての機能に加え、データを暗号化するオプションが提供されます。詳細については、次を参照してください:

- [ElastiCache 転送時の暗号化 \(TLS\)](#)
- [ElastiCache での保管時の暗号化](#)
- [Amazon のコンプライアンス検証 ElastiCache](#)

ElastiCache for Redis バージョン 3.2.4 (拡張)

Amazon ElastiCache for Redis バージョン 3.2.4 で、Amazon ElastiCache でサポートされている Redis エンジンの次の主要バージョンが導入されます。ElastiCache for Redis 3.2.4 のユーザーには、以前のバージョンの Redis で使用できるすべての機能に加え、クラスターモードまたは非クラスターモードで実行するオプションも提供されます。次の表に以下の内容がまとめてあります。

Redis 3.2.4 非クラスターモードとクラスターモードの比較

機能	非クラスターモード	クラスターモード
データのパーティション化	いいえ	はい
地理空間インデックス作成	はい	はい
ノードタイプの変更	はい	はい *
レプリカの拡張	はい	はい *
スケールアウト	いいえ	はい *
データベースのサポート	複数	単一
パラメータグループ	default.redis3.2 **	default.redis3.2.cluster.on **

* 「[バックアップから新しいキャッシュへの復元](#)」を参照してください

** またはそれから派生したもの。

注記：

- [パーティション] – データを 2~500 のノードグループ間で分割 (シャード) (各ノードグループでレプリケーションのサポートあり)。
- [地理空間インデックス作成] – Redis 3.2.4 で、6 つの GEO コマンドによる地理空間インデックス作成のサポートが追加されました。詳細については、Redis GEO* コマンドのドキュメントの Redis コマンドページ (GEO でフィルタリング) の「[Redis コマンド: GEO](#)」を参照してください。

Redis 3 の追加の機能については、『[Redis 3.2 リリースノート](#)』と『[Redis 3.0 リリースノート](#)』を参照してください。

現在、ElastiCache マネージド Redis (クラスターモードが有効) では、Redis 3.2 の以下の機能はサポートされていません。

- レプリカの移行
- クラスターの再分散
- Lua デバッガー

ElastiCacheでは、Redis 3.2 の以下の管理コマンドは無効になっています。

- `cluster meet`
- `cluster replicate`
- `cluster flushslots`
- `cluster addslots`
- `cluster delslots`
- `cluster setslot`
- `cluster saveconfig`
- `cluster forget`
- `cluster failover`
- `cluster bumpepoch`
- `cluster set-config-epoch`
- `cluster reset`

Redis 3.2.4 のパラメータの詳細については、『[Redis 3.2.4 パラメータの変更](#)』を参照してください。

過去のサポート終了 (EOL) バージョン (2.x)

ElastiCache for Redis バージョン 2.8.24 (拡張)

バージョン 2.8.23 以降に追加された、バグ修正および不正なメモリアクセスのアドレスのログ記録を含む Redis の改善 詳細については、『[Redis 2.8 リリースノート](#)』を参照してください。

ElastiCache for Redis バージョン 2.8.23 (拡張)

バージョン 2.8.22 以降に追加された Redis の機能拡張には、バグ修正も含まれます。詳細については、「[Redis 2.8 リリースノート](#)」を参照してください。また、このリリースでは、新しいパラメータ `close-on-slave-write` もサポートされており、有効にした場合、読み取り専用レプリカに書き込もうとするクライアントの接続は切断されます。

Redis 2.8.23 のパラメータの詳細については、ElastiCache ユーザーガイドの「[Redis 2.8.23 \(拡張\) で追加されたパラメータ](#)」を参照してください。

ElastiCache for Redis バージョン 2.8.22 (拡張)

バージョン 2.8.21 以降に追加された Redis の機能拡張には、以下が含まれます。

- 分岐なしのバックアップと同期のサポートにより、バックアップオーバーヘッドによるメモリの割り当てを減らしてより多くのメモリをアプリケーションに割り当てることができます。詳細については、「[同期とバックアップの実装方法](#)」を参照してください。分岐なしのプロセスは、レイテンシーとスループットの両方に影響を与える場合があります。書き込みスループットが高い場合、レプリカが再同期されると、同期中はレプリカにアクセスできなくなることがあります。
- フェイルオーバーが発生した場合、可能であれば、レプリカがフル同期ではなくプライマリとの部分同期を実行するため、レプリケーショングループはより迅速に復旧されます。さらに、プライマリとレプリカは同期中にディスクを使用しないため、速度が向上します。
- が新しい CloudWatch メトリクスをサポート
 - ReplicationBytes – レプリケーショングループのプライマリクラスターがリードレプリカに送信しているバイト数。
 - SaveInProgress – バックグラウンド保存プロセスが実行されるかどうかを示すバイナリ値。

詳細については、「[CloudWatch メトリクスの使用状況のモニタリング](#)」を参照してください。

- レプリケーション PSYNC 動作のいくつかの重要なバグ修正。詳細については、「[Redis 2.8 リリースノート](#)」を参照してください。
- マルチ AZ レプリケーショングループのレプリケーションパフォーマンスの拡張とクラスターの安定性を維持するために、非 ElastiCache レプリカのサポートが終了しました。
- レプリケーショングループのプライマリクラスターとレプリカ間でデータの整合性を改善するために、プライマリクラスターと無関係にレプリカでキーを削除できなくなりました。
- Redis 構成変数 `appendonly` および `appendfsync` Redis バージョン 2.8.22 以降ではサポートされていません。

- メモリが少ない状況で、大きな出力アップロードバッファを持つクライアントはレプリカクラスターからの接続が解除されることがあります。接続が解除された場合、クライアントは再接続する必要があります。このような状況は、多くの場合 PUBSUB クライアントで発生する可能性があります。

ElastiCache for Redis バージョン 2.8.21

Redis のバージョン 2.8.19 で数多くのバグ修正が行われ、改善されました。詳細については、「[Redis 2.8 リリースノート](#)」を参照してください。

ElastiCache for Redis バージョン 2.8.19

バージョン 2.8.6 以降に追加された Redis の機能拡張には、以下が含まれます。

- HyperLogLog のサポート。詳細については、「[Redis の新しいデータ構造: HyperLogLog](#)」を参照してください。
- ソートされたセットデータ型は、新しいコマンド ZRANGEBYLEX、ZLEXCOUNT、および ZREMRANGEBYLEX で、辞書式範囲のクエリをサポートできるようになりました。
- プライマリノードがレプリカノードに古いデータを送信しないようにするため、バックグラウンド保存 (bgsave) の子プロセスが中止された場合、マスター SYNC は失敗します。
- HyperLogLogBasedCommands CloudWatch メトリクスがサポートされました。詳細については、「[Redis のメトリクス](#)」を参照してください。

ElastiCache for Redis バージョン 2.8.6

バージョン 2.6.13 以降に追加された Redis の機能拡張には、以下が含まれます。

- リードレプリカの弾力性と耐障害性の向上。
- 部分的な再同期のサポート。
- 常に使用できる必要があるリードレプリカの最小数に関するユーザー定義のサポート。
- pub/sub のフルサポートサーバー上のイベントをクライアントに通知。
- プライマリノードの障害の自動検出と、プライマリノードからセカンダリノードへのフェイルオーバー。

ElastiCache for Redis バージョン 2.6.13

Redis バージョン 2.6.13 は、Amazon ElastiCache for Redis でサポートされた Redis の最初のバージョンです。マルチ AZ は Redis 2.6.13 ではサポートされません。

Redis バージョンのサポート終了スケジュール

このセクションでは、発表された以前のメジャーバージョンのサポート終了 (EOL、End Of Life) 日を定義します。これにより、将来のバージョンとアップグレードに関する判断を行うことができます。

Note

5.0.0 から 5.0.5 までの ElastiCache for Redis パッチバージョンは廃止されました。5.0.6 以降のバージョンを使用してください。

次の表は、各バージョンと、その発表された EOL 日、および推奨されるアップグレードターゲットバージョンをまとめたものです。

過去の EOL

ソースマイナーバージョン	推奨アップグレードターゲット	EOL 日
3.2.4、3.2.6、3.2.10	バージョン 6.2 以上	2023 年 7 月 31 日
	 Note US-ISO-EA ST-1、US-ISO-WEST-1、および US-ISOB-EAST-1 リー	

ソースマイナーバージョン	推奨アップグレードターゲット	EOL 日
	<p>ジョンでは、5.0.6 以上をお勧めします。</p>	
<p>2.8.24、2.8.23、2.8.22、2.8.21、2.8.19、2.8.12、2.8.6、2.6.13</p>	<p>バージョン 6.2 以上</p> <p>Note US-ISO-EA、ST-1、US-ISO-WEST-1、および US-ISOB-EAST-1 リージョンでは、5.0.6 以上をお勧めします。</p>	<p>2023 年 1 月 13 日</p>

エンジンバージョンのアップグレード方法

クラスターまたはレプリケーショングループのバージョンアップは、ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、より新しいエンジンバージョンを指定することで開始します。詳細については、以下のトピックを参照してください。

クラスターおよびレプリケーショングループを変更する方法

クラスター	レプリケーショングループ
の使用 AWS Management Console	の使用 AWS Management Console
の使用 AWS CLI	の使用 AWS CLI
ElastiCache API の使用	ElastiCache API の使用

ブロックされた Redis エンジンのアップグレードの解決

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

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

ブロックされた Redis エンジンのアップグレードを解決するには

- 次のいずれかを行います:
 - [すぐに適用] チェックボックスをオフにすることで、Redis エンジンのアップグレードオペレーションを次のメンテナンスウィンドウに予定します。

CLI では、`--no-apply-immediately` を使用します。API では、`ApplyImmediately=false` を使用します。

- Redis のエンジンアップグレードオペレーションを実行する次のメンテナンス期間 (またはその後) まで待ちます。
- [すぐに適用] チェックボックスをオンにすることで、Redis のスケールアップオペレーションをこのクラスターの変更に追加します。

CLI では、`--apply-immediately` を使用します。API では、`ApplyImmediately=true` を使用します。

このアプローチにより、エンジンのアップグレードがすぐに実行されて、次のメンテナンスウィンドウ中のエンジンのアップグレードはキャンセルされます。

メジャーバージョンの動作と互換性の違い

Important

次のページは、バージョン間の非互換性の違いをすべて示し、新しいバージョンにアップグレードする際に考慮すべき事項を説明するように構成されています。このリストには、アップグレード時に発生する可能性のある、バージョンの非互換性についての問題が含まれています。

現在の Redis バージョンから利用可能な最新の Redis バージョンに直接アップグレードでき、順次アップグレードする必要はありません。例えば、Redis バージョン 3.0 からバージョン 7.0 に直接アップグレードできます。

Redis のバージョンは、メジャー、マイナー、およびパッチコンポーネントで構成されるセマンティックバージョンで識別されます。例えば、Redis 4.0.10 では、メジャーバージョンは 4、マイナーバージョンは 0、パッチバージョンは 10 です。これらの値は、通常、次の規則に基づいて増分されます。

- メジャーバージョンは API 非互換の変更に用います
- マイナーバージョンは、下位互換性のある方法で追加された新機能に用います
- パッチバージョンは、下位互換性のあるバグ修正と機能以外の変更に用います

最新のパフォーマンスと安定性の向上のために、特定の MAJOR.MINOR バージョン内の最新のパッチバージョンを常に使用することをお勧めします。ElastiCache for Redis では、Redis 6.0 以降、複数のパッチバージョンを提供するのではなく、Redis OSS マイナーリリースごとに 1 つのバージョンが提供されます。ElastiCache for Redis は、実行中のキャッシュクラスターのパッチバージョンを自動的に管理し、パフォーマンスの向上とセキュリティ強化を実現します。

また、ほとんどの主要な改善が古いバージョンにバックポートされないため、最新のメジャーバージョンに定期的にアップグレードすることをお勧めします。ElastiCache が別の AWS リージョンでも利用可能になると、そのリージョンでは、その時点で最新の 2 つのメジャーバージョンとマイナーバージョンの ElastiCache for Redis がサポートされます。例えば、ElastiCache for Redis が別の AWS リージョンで利用可能になり、最新のメジャーバージョンとマイナーバージョンが 7.0 と 6.2 である場合、その AWS リージョンでは、バージョン 7.0 と 6.2 の ElastiCache for Redis がサポートされます。ElastiCache for Redis の新しいメジャーバージョンとマイナーバージョンがリリースされるたびに、新しくリリースされたバージョンの ElastiCache for Redis へのサポートが追加されます。ElastiCache のリージョンの選択について詳しくは、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

複数のメジャーバージョンまたはマイナーバージョンにまたがるアップグレードを行う場合は、次のリストを考慮してください。このリストには、時間の経過と共に Redis でリリースされる動作と下位互換性のない変更が含まれています。

Redis 7.0 の動作および下位互換性のない変更

変更のリストについては、「[Redis 7.0 リリースノート](#)」を参照してください。

- SCRIPT LOAD と SCRIPT FLUSH はレプリカに伝播されなくなります。スクリプトをある程度耐久性が必要な場合は、[Redis 関数](#)の使用を検討することをお勧めします。
- Pubsub チャンネルは、新しい ACL ユーザーにはデフォルトでブロックされるようになりました。
- STRALGO コマンドは LCS コマンドに置き換えられました。
- ACL GETUSER の形式が変更され、すべてのフィールドに標準のアクセス文字列パターンが表示されるようになりました。ACL GETUSER を使用してオートメーションを行った場合は、どちらの形式でも処理できることを確認する必要があります。
- SELECT、WAIT、ROLE、LASTSAVE、READONLY、READWRITE、ASKING の ACL カテゴリが変更されました。
- INFO コマンドは、トップレベルのコンテナコマンドではなく、サブコマンドごとのコマンド統計を表示するようになりました。

- LPOP、RPOP、ZPOPMIN、ZPOPMAX コマンドの戻り値が特定のエッジケースで変更されました。これらのコマンドを使用する場合は、リリースノートを確認して、影響を受けるかどうかを評価する必要があります。
- SORT および SORT_RO コマンドで GET および BY 引数を使用するためには、キースペース全体へのアクセスが必要になりました。

Redis 6.2 の動作および下位互換性のない変更

変更の詳細なリストについては、「[Redis 6.2 リリースノート](#)」を参照してください。

- TIME、ECHO、ROLE、および LASTSAVE コマンドの ACL フラグが変更されました。これにより、以前は許可されていたコマンドが拒否されたり、その逆のことが起こったりする可能性があります。

Note

これらのコマンドはいずれも、データを変更したり、データにアクセスしたりすることはありません。

- Redis 6.0 からアップグレードすると、lua スクリプトへのマップ応答から返されるキーと値のペアの順序が変更されます。スクリプトが `redis.setresp()` を使用するが、マップ (Redis 6.0 の新機能) を返す場合は、スクリプトがアップグレード時に中断する可能性がある影響を考慮してください。

Redis 6.0 の動作および下位互換性のない変更

変更の詳細なリストについては、「[Redis 6.0 リリースノート](#)」を参照してください。

- 許可されるデータベースの最大数は 120 万から 1 万に減少しました。デフォルト値は 16 です。パフォーマンスとメモリの懸念が見つかったため、これよりはるかに大きい値の使用はお勧めしません。
- `AutoMinorVersionUpgrade` パラメータを `yes` に設定すると、ElastiCache for Redis はセルフサービスの更新によってマイナーバージョンのアップグレードを管理します。これは、セルフサービス更新キャンペーンを通じて、標準的な顧客通知チャネルを通じて処理されます。詳細については、「[ElastiCache でのセルフサービスの更新](#)」をご覧ください。

Redis 5.0 の動作および下位互換性のない変更

変更の詳細なリストについては、「[Redis 5.0 リリースノート](#)」を参照してください。

- スクリプトは、レプリカでスクリプトを再実行するのではなく、効果によってレプリケートされます。これにより、一般にパフォーマンスは向上しますが、プライマリとレプリカの間でレプリケートされるデータ量が増える可能性があります。ElastiCache for Redis 5.0 でのみ使用可能な、以前の動作に戻すオプションがあります。
- Redis 4.0 からアップグレードする場合、LUA スクリプトの一部のコマンドは、以前のバージョンとは異なる順序で引数を返します。Redis 4.0 では、Redis は応答を確定的にするためにいくつかの応答を辞書順に並べますが、スクリプトが効果によって複製される場合、この順序は適用されません。
- Redis 5.0.3 以降、ElastiCache for Redis は、いくつかの IO 作業を 4 個以上の VCPU を持つインスタンスタイプのバックグラウンドコアにオフロードします。これにより、Redis のパフォーマンス特性が変更され、一部のメトリクスの値が変更される可能性があります。詳細については、「[モニタリングすべきメトリクス](#)」を参照して、監視するメトリクスを変更する必要があるかどうかを理解してください。

Redis 4.0 の動作および下位互換性のない変更

変更のリストについては、「[Redis 4.0 リリースノート](#)」を参照してください。

- スローログは、クライアント名とアドレスという追加の 2 つの引数をログに記録するようになりました。この変更は、3 つの値を含む各スローログエントリに明示的に依存しない限り、下位互換性があります。
- CLUSTER NODES コマンドは、わずかに異なる形式を返すようになりましたが、これには下位互換性がありません。クライアントは、クラスタ内に存在するノードについて学習するためにこのコマンドを使用するのではなく、CLUSTER SLOTS を使用することをお勧めします。

過去の EOL

Redis 3.2 の動作および下位互換性のない変更

変更の詳細なリストについては、「[Redis 3.2 リリースノート](#)」を参照してください。

- このバージョンでは、注意すべき互換性の変更はありません。

詳細については、「[Redis バージョンのサポート終了スケジュール](#)」を参照してください。

Redis 2.8 の動作および下位互換性のない変更

変更の詳細なリストについては、「[Redis 2.8 リリースノート](#)」を参照してください。

- Redis 2.8.22 以降、Redis AOF は、ElastiCache for Redis ではサポートされなくなりました。データを永続的に保持する必要がある場合は、MemoryDB を使用することをお勧めします。
- Redis 2.8.22 以降、ElastiCache for Redis は、ElastiCache 内でホストされているプライマリへのレプリカのアタッチをサポートしなくなりました。アップグレード中、外部レプリカは切断され、再接続できなくなります。外部レプリカの代わりに Redis 6.0 で利用できるクライアント側のキャッシュを使用することをお勧めします。
- TTL および PTTL コマンドは、キーが存在しない場合は -2 を返し、存在しても関連する有効期限がない場合は -1 を返すようになりました。Redis 2.6 以前のバージョンでは、両方の条件で -1 を返していました。
- STORE オプションが使用されていない場合、ALPHA 付きの SORT はローカル照合ロケールに従ってソートされるようになりました。

詳細については、「[Redis バージョンのサポート終了スケジュール](#)」を参照してください。

ElastiCache ベストプラクティスとキャッシュ戦略

Amazon の推奨ベストプラクティスを以下に示します ElastiCache。これらに従うと、キャッシュのパフォーマンスと信頼性が向上します。

トピック

- [Redis の使用](#)
- [Redis クライアントに関するベストプラクティス](#)
- [予約メモリの管理](#)
- [独自設計型クラスターを使用する場合のベストプラクティス](#)
- [Redis のベストプラクティス](#)
- [キャッシュ戦略](#)

Redis の使用

以下に、ElastiCache 内の Redis インターフェースに関する情報があります。

トピック

- [サポートされている Redis コマンドと制限されている Redis コマンド](#)
- [Redis の設定と制限事項](#)

サポートされている Redis コマンドと制限されている Redis コマンド

サポートされている Redis コマンド

サポートされている Redis コマンド

次の Redis コマンドはサーバーレスキャッシュでサポートされています。これらのコマンドに加えて、これらの [サポートされている Redis JSON コマンド](#) もサポートされています。

ビットマップコマンド

- BITCOUNT

文字列内の設定されているビット数をカウントします (母集団計数)。

[詳細はこちら](#)

- BITFIELD

文字列に対して任意のビットフィールド整数演算を実行します。

[詳細はこちら](#)

- BITFIELD_RO

文字列に対して任意の読み取り専用ビットフィールド整数演算を実行します。

[詳細はこちら](#)

- BITOP

複数の文字列に対してビット演算を行い、結果を格納します。

[詳細はこちら](#)

- BITPOS

文字列の最初の設定されている (1) ビットまたはクリア (0) ビットを検索します。

[詳細はこちら](#)

- GETBIT

ビット値をオフセットで返します。

[詳細はこちら](#)

- SETBIT

文字列値のオフセットのビットを設定またはクリアします。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

クラスター管理コマンド

- CLUSTER COUNTKEYSINSLOT

ハッシュスロット内のキーの数を返します。

[詳細はこちら](#)

- CLUSTER GETKEYSINSLOT

ハッシュスロット内のキーの名前を返します。

[詳細はこちら](#)

- CLUSTER INFO

ノードの状態に関する情報を返します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- CLUSTER KEYSLOT

キーのハッシュスロットを返します。

[詳細はこちら](#)

- CLUSTER MYID

ノードの ID を返します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- CLUSTER NODES

ノードのクラスター構成を返します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- CLUSTER REPLICAS

マスターノードのレプリカノードを一覧表示します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- CLUSTER SHARDS

クラスタースロットとシャードのマッピングを返します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- CLUSTER SLOTS

クラスタースロットとノードのマッピングを返します。サーバーレスキャッシュでは、クライアントに公開されている 1 つの仮想「シャード」に関する状態を返します。

[詳細はこちら](#)

- READONLY

Redis Cluster レプリカノードへの接続に対する読み取り専用クエリを有効にします。

[詳細はこちら](#)

- READWRITE

Redis Cluster レプリカノードへの接続に対する読み書きクエリを有効にします。

[詳細はこちら](#)

接続管理コマンド

- AUTH

接続を認証します。

[詳細はこちら](#)

- CLIENT GETNAME

接続の名前を返します。

[詳細はこちら](#)

- CLIENT REPLY

コマンドに応答するかどうかをサーバーに指示します。

[詳細はこちら](#)

- CLIENT SETNAME

接続名を設定します。

[詳細はこちら](#)

- ECHO

指定された文字列を返します。

[詳細はこちら](#)

- HELLO

Redis サーバーとハンドシェイクします。

[詳細はこちら](#)

- PING

サーバーのライブネスレスポンスを返します。

[詳細はこちら](#)

- QUIT

接続を閉じます。

[詳細はこちら](#)

- RESET

接続をリセットします。

[詳細はこちら](#)

- SELECT

選択したデータベースを変更します。

[詳細はこちら](#)

汎用コマンド

- COPY

キーの値を新しいキーにコピーします。

[詳細はこちら](#)

- DEL

1 つまたは複数のキーを削除します。

[詳細はこちら](#)

- DUMP

キーに格納されている値をシリアル化して返します。

[詳細はこちら](#)

- EXISTS

1 つまたは複数のキーが存在するかどうかを判定します。

[詳細はこちら](#)

- EXPIRE

キーの有効期限を秒単位で設定します。

[詳細はこちら](#)

- EXPIREAT

キーの有効期限を Unix タイムスタンプに設定します。

[詳細はこちら](#)

- EXPIRETIME

キーの有効期限を Unix タイムスタンプとして返します。

[詳細はこちら](#)

- PERSIST

キーの有効期限を削除します。

[詳細はこちら](#)

- PEXPIRE

キーの有効期限をミリ秒単位で設定します。

[詳細はこちら](#)

- PEXPIREAT

キーの有効期限を Unix タイムスタンプに設定します。

[詳細はこちら](#)

- PEXPIRETIME

キーの有効期限を Unix ミリ秒タイムスタンプとして返します。

[詳細はこちら](#)

- PTTL

キーのミリ秒単位の有効期限を返します。

[詳細はこちら](#)

- RANDOMKEY

データベースからランダムなキー名を返します。

[詳細はこちら](#)

- RENAME

キーの名前を変更し、変更先を上書きします。

[詳細はこちら](#)

- RENAMENX

ターゲットのキー名が存在しない場合にのみキーの名前を変更します。

[詳細はこちら](#)

- RESTORE

シリアル化された値の表現からキーを作成します。

[詳細はこちら](#)

- SCAN

データベース内のキー名を繰り返し処理します。

[詳細はこちら](#)

- SORT

リスト、セット、またはソートセット内の要素をソートし、オプションで結果を格納します。

[詳細はこちら](#)

- SORT_RO

リスト、セット、またはソートされたセットのソートされた要素を返します。

[詳細はこちら](#)

- TOUCH

指定したキーのうち、最後にアクセスされた時刻を更新したキーの数を返します。

[詳細はこちら](#)

- TTL

キーの秒単位の有効期限を返します。

[詳細はこちら](#)

- TYPE

キーに格納されている値のタイプを決定します。

[詳細はこちら](#)

- UNLINK

1 つまたは複数のキーを非同期的に削除します。

[詳細はこちら](#)

地理空間コマンド

- GEOADD

地理空間インデックスに 1 つまたは複数のメンバーを追加します。キーが存在しない場合はキーが作成されます。

[詳細はこちら](#)

- GEODIST

地理空間インデックスの 2 つのメンバー間の距離を返します。

[詳細はこちら](#)

- GEOHASH

地理空間インデックスのメンバーをジオハッシュ文字列として返します。

[詳細はこちら](#)

- GEOPOS

地理空間インデックスからメンバーの経度と緯度を返します。

[詳細はこちら](#)

- GEORADIUS

指定した座標から特定の距離内にあるメンバーの地理空間インデックスを照会し、オプションで結果を格納します。

[詳細はこちら](#)

- GEORADIUS_R0

地理空間インデックスから、指定した座標から特定の距離内にあるメンバーを返します。

[詳細はこちら](#)

- GEORADIUSBYMEMBER

指定したメンバーから特定の距離内にあるメンバーの地理空間インデックスを照会し、オプションで結果を格納します。

[詳細はこちら](#)

- GEORADIUSBYMEMBER_RO

地理空間インデックスから、指定したメンバーから特定の距離内にあるメンバーを返します。

[詳細はこちら](#)

- GEOSEARCH

ボックスまたは円の領域内のメンバーの地理空間インデックスを照会します。

[詳細はこちら](#)

- GEOSEARCHSTORE

ボックスまたは円の領域内のメンバーの地理空間インデックスを照会し、オプションで結果を格納します。

[詳細はこちら](#)

ハッシュコマンド

- HDEL

1つまたは複数のフィールドとその値をハッシュから削除します。フィールドが残っていない場合はハッシュを削除します。

[詳細はこちら](#)

- HEXISTS

ハッシュにフィールドが存在するかどうかを判定します。

[詳細はこちら](#)

- HGET

ハッシュ内のフィールドの値を返します。

[詳細はこちら](#)

- HGETALL

ハッシュ内のすべてのフィールドと値を返します。

[詳細はこちら](#)

- HINCRBY

ハッシュ内のフィールドの整数値を数値分増やします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- HINCRBYFLOAT

ハッシュ内のフィールドの浮動小数点値を数値分増やします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- HKEYS

ハッシュ内のすべてのフィールドを返します。

[詳細はこちら](#)

- HLEN

ハッシュ内のフィールドの数を返します。

[詳細はこちら](#)

- HMGET

ハッシュ内のすべてのフィールドを返します。

[詳細はこちら](#)

- HMSET

複数のフィールドの値を設定します。

[詳細はこちら](#)

- HRANDFIELD

ハッシュから 1 つまたは複数のランダムフィールドを返します。

[詳細はこちら](#)

- HSCAN

ハッシュのフィールドと値を反復処理します。

[詳細はこちら](#)

- HSET

ハッシュ内のフィールドの値を作成または変更します。

[詳細はこちら](#)

- HSETNX

フィールドが存在しない場合にのみ、ハッシュ内のフィールドの値を設定します。

[詳細はこちら](#)

- HSTRLEN

フィールドの値の長さを返します。

[詳細はこちら](#)

- HVALS

ハッシュ内のすべての値を返します。

[詳細はこちら](#)

HyperLogLog コマンド

- PFADD

HyperLogLog キーに要素を追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- PFCOUNT

HyperLogLog キーで観測されたセットのおおよそのカーディナリティを返します。

[詳細はこちら](#)

- PFMERGE

1 つまたは複数の HyperLogLog 値を 1 つのキーにマージします。

[詳細はこちら](#)

リストコマンド

- BLMOVE

リストから要素を取り出し、別のリストにプッシュして返します。要素が利用可能になるまでブロックします。最後の要素を移動した場合はリストを削除します。

[詳細はこちら](#)

- BLMPOP

複数のリストのうちの 1 つから最初の要素を取り出します。要素が利用可能になるまでブロックします。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- BLPOP

リストの最初の要素を削除して返します。要素が利用可能になるまでブロックします。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- BRPOP

リストの最後の要素を削除して返します。要素が利用可能になるまでブロックします。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- BRPOPLPUSH

リストから要素を取り出し、別のリストにプッシュして返します。要素が利用可能になるまでブロックします。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- LINDEX

リストの要素をインデックスで返します。

[詳細はこちら](#)

- LINSERT

リスト内の別の要素の前または後に要素を挿入します。

[詳細はこちら](#)

- LLEN

リストの長さを返します。

[詳細はこちら](#)

- LMOVE

あるリストから取り出して別のリストにプッシュした要素を返します。最後の要素を移動した場合はリストを削除します。

[詳細はこちら](#)

- LMPOP

リストから複数の要素を削除して返します。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- LPOP

リストの最初の要素を削除して返します。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- LPOS

リスト内の一致する要素のインデックスを返します。

[詳細はこちら](#)

- LPUSH

1 つまたは複数の要素をリストの先頭に追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- LPUSHX

リストが存在する場合のみ、1 つまたは複数の要素をリストの先頭に追加します。

[詳細はこちら](#)

- LRANGE

リストから一定範囲の要素を返します。

[詳細はこちら](#)

- LREM

リストから要素を削除します。最後の要素を削除した場合はリストを削除します。

[詳細はこちら](#)

- LSET

リスト内の要素の値をインデックスで設定します。

[詳細はこちら](#)

- LTRIM

リストの両端から要素を削除します。すべての要素をトリムした場合はリストを削除します。

[詳細はこちら](#)

- RPOP

リストの最後の要素を削除して返します。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- RPOPLPUSH

リストの最後の要素を削除して別のリストにプッシュした後、その要素を返します。最後の要素を取り出した場合はリストを削除します。

[詳細はこちら](#)

- RPUSH

1 つまたは複数の要素をリストに追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- RPUSHX

リストが存在する場合のみ、リストに要素を追加します。

[詳細はこちら](#)

Pub/Sub コマンド

Note

PUBSUB コマンドはシャード化された PUBSUB を内部的に使用するため、チャンネル名は混在することになります。

- PUBLISH

チャンネルにメッセージを投稿します。

[詳細はこちら](#)

- PUBSUB CHANNELS

アクティブなチャンネルを返します。

[詳細はこちら](#)

- PUBSUB NUMSUB

チャンネルのサブスクライバーを返します。

[詳細はこちら](#)

- PUBSUB SHARDCHANNELS

アクティブなシャードチャンネルを返します。

[PUBSUB-SHARDCHANNELS](#)

- PUBSUB SHARDNUMSUB

シャードチャンネルのサブスクライバー数を返します。

[PUBSUB-SHARDNUMSUB](#)

- SPUBLISH

シャードチャンネルにメッセージを投稿します。

[詳細はこちら](#)

- SSUBSCRIBE

シャードチャンネルに公開されたメッセージをリスニングします。

[詳細はこちら](#)

- SUBSCRIBE

チャンネルに公開されたメッセージをリスニングします。

[詳細はこちら](#)

- SUNSUBSCRIBE

シャードチャンネルに投稿されたメッセージのリスニングを停止します。

[詳細はこちら](#)

- UNSUBSCRIBE

チャンネルに投稿されたメッセージのリスニングを停止します。

[詳細はこちら](#)

スクリプトコマンド

- EVAL

サーバー側 Lua スクリプトを実行します。

[詳細はこちら](#)

- EVAL_R0

読み取り専用のサーバー側 Lua スクリプトを実行します。

[詳細はこちら](#)

- EVALSHA

サーバー側 Lua スクリプトを SHA1 ダイジェストで実行します。

[詳細はこちら](#)

- EVALSHA_R0

読み取り専用のサーバー側 Lua スクリプトを SHA1 ダイジェストで実行します。

[詳細はこちら](#)

- SCRIPT EXISTS

サーバー側 Lua スクリプトがスクリプトキャッシュに存在するかどうかを判定します。

[詳細はこちら](#)

- SCRIPT FLUSH

現在、運用は不要のスクリプトキャッシュがサービスによって管理されています。

[詳細はこちら](#)

- SCRIPT LOAD

サーバー側の Lua スクリプトをスクリプトキャッシュに読み込みます。

[詳細はこちら](#)

サーバー管理コマンド

- ACL CAT

ACL カテゴリ、またはカテゴリ内のコマンドを一覧表示します。

[詳細はこちら](#)

- ACL GENPASS

ACL ユーザーの識別に使用できる疑似ランダムで安全なパスワードを生成します。

[詳細はこちら](#)

- ACL GETUSER

ユーザーの ACL ルールを一覧表示します。

[詳細はこちら](#)

- ACL LIST

有効なルールを ACL ファイル形式でダンプします。

[詳細はこちら](#)

- ACL USERS

すべての ACL ユーザーを一覧表示します。

[詳細はこちら](#)

- ACL WHOAMI

現在の接続の認証済みユーザー名を返します。

[詳細はこちら](#)

- DBSIZE

現在選択されているデータベース内のキーの数を返します。この操作がすべてのスロットでアトミックになるとは限りません。

[詳細はこちら](#)

- COMMAND

すべてのコマンドに関する詳細情報を返します。

[詳細はこちら](#)

- COMMAND COUNT

コマンドの数を返します。

[詳細はこちら](#)

- COMMAND DOCS

1 つ、複数、またはすべてのコマンドに関するドキュメンタリー情報を返します。

[詳細はこちら](#)

- COMMAND GETKEYS

任意のコマンドからキー名を抽出します。

[詳細はこちら](#)

- COMMAND GETKEYSANDFLAGS

任意のコマンドのキー名とアクセスフラグを抽出します。

[詳細はこちら](#)

- COMMAND INFO

1 つ、複数、またはすべてのコマンドに関する情報を返します。

[詳細はこちら](#)

- COMMAND LIST

コマンド名のリストを返します。

[詳細はこちら](#)

- FLUSHALL

すべてのデータベースからすべてのキーを削除します。この操作がすべてのスロットでアトミックになるとは限りません。

[詳細はこちら](#)

- FLUSHDB

現在のデータベースからすべてのキーを削除します。この操作がすべてのスロットでアトミックになるとは限りません。

[詳細はこちら](#)

- INFO

サーバーに関する情報と統計を返します。

[詳細はこちら](#)

- LOLWUT

コンピューターアートと Redis バージョンを表示します。

[詳細はこちら](#)

- ROLE

レプリケーションロールを返します。

[詳細はこちら](#)

- TIME

サーバー時間を返します。

[詳細はこちら](#)

集合コマンド

- SADD

集合に 1 つまたは複数のメンバーを追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- SCARDT

集合内のメンバー数を返します。

[詳細はこちら](#)

- SDIFF

複数の集合の差を返します。

[詳細はこちら](#)

- SDIFFSTORE

複数の集合の差をキーに格納します。

[詳細はこちら](#)

- SINTER

複数の集合の交差を返します。

[詳細はこちら](#)

- SINTERCARD

複数の集合の交差のメンバー数を返します。

[詳細はこちら](#)

- SINTERSTORE

複数の集合の交差をキーに格納します。

[詳細はこちら](#)

- SISMEMBER

メンバーが集合に属しているかどうかを判定します。

[詳細はこちら](#)

- SMEMBERS

集合のすべてのメンバーを返します。

[詳細はこちら](#)

- SMISMEMBER

複数のメンバーが集合に属しているかどうかを判定します。

[詳細はこちら](#)

- SMOVE

メンバーをある集合から別の集合に移動します。

[詳細はこちら](#)

- SPOP

集合か 1 つまたは複数のメンバーをランダムに削除して返します。最後のメンバーを取り出した場合は集合を削除します。

[詳細はこちら](#)

- SRANDMEMBER

集合から 1 つまたは複数のメンバーをランダムに取得します。

[詳細はこちら](#)

- SREM

集合から 1 つまたは複数のメンバーを削除します。最後のメンバーを削除した場合は集合を削除します。

[詳細はこちら](#)

- SSCAN

集合のメンバーを反復処理します。

[詳細はこちら](#)

- SUNION

複数の集合の和を返します。

[詳細はこちら](#)

- SUNIONSTORE

複数の集合の和をキーに格納します。

[詳細はこちら](#)

ソート対象集合コマンド

- BZMPOP

1 つまたは複数のソート対象集合からメンバーをスコアによって削除して返します。メンバーが使用可能になるまでブロックします。最後の要素を取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- BZPOPMAX

1 つまたは複数のソート対象集合からスコアの最も高いメンバーを削除して返します。メンバーが使用可能になるまでブロックします。最後の要素を取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- BZPOPMIN

1 つまたは複数のソート対象集合からスコアの最も低いメンバーを削除して返します。メンバーが使用可能になるまでブロックします。最後の要素を取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- ZADD

ソート対象集合に 1 つまたは複数のメンバーを追加するか、メンバーのスコアを更新します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- ZCARD

ソート対象集合内のメンバー数を返します。

[詳細はこちら](#)

- ZCOUNT

ソート対象集合の中で、スコアが範囲内にあるメンバーの数を返します。

[詳細はこちら](#)

- ZDIFF

複数のソート対象集合間の差を返します。

[詳細はこちら](#)

- ZDIFFSTORE

複数のソート対象集合の差をキーに格納します。

[詳細はこちら](#)

- ZINCRBY

ソート対象集合内のメンバーのスコアをインクリメントします。

[詳細はこちら](#)

- ZINTER

複数のソート対象集合の交差を返します。

[詳細はこちら](#)

- ZINTERCARD

複数のソート対象集合の交差のメンバー数を返します。

[詳細はこちら](#)

- ZINTERSTORE

複数のソート対象集合の交差をキーに格納します。

[詳細はこちら](#)

- ZLEXCOUNT

ソート対象集合の辞書的範囲内のメンバー数を返します。

[詳細はこちら](#)

- ZMPOP

1つまたは複数のソート対象集合から最高スコアまたは最低スコアのメンバーを削除して返します。最後のメンバーを取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- ZMSCORE

ソート対象集合内の1つまたは複数のメンバーのスコアを返します。

[詳細はこちら](#)

- ZPOPMAX

1つまたは複数のソート対象集合から最高スコアのメンバーを削除して返します。最後のメンバーを取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- ZPOPMIN

1 つまたは複数のソート対象集合から最低スコアのメンバーを削除して返します。最後のメンバーを取り出した場合はソート対象集合を削除します。

[詳細はこちら](#)

- ZRANDMEMBER

ソート対象集合から 1 つまたは複数のランダムなメンバーを返します。

[詳細はこちら](#)

- ZRANGE

ソート対象集合のインデックス範囲内のメンバーを返します。

[詳細はこちら](#)

- ZRANGEBYLEX

ソート対象集合の辞書的範囲内のメンバーを返します。

[詳細はこちら](#)

- ZRANGEBYSCORE

スコアの範囲内にあるソート対象集合のメンバーを返します。

[詳細はこちら](#)

- ZRANGESTORE

ソート対象集合のメンバーの範囲をキーに格納します。

[詳細はこちら](#)

- ZRANK

ソート対象集合内のメンバーのインデックスをスコアの昇順で返します。

[詳細はこちら](#)

- ZREM

ソート対象集合から 1 つまたは複数のメンバーを削除します。すべてのメンバーを削除した場合、ソート対象集合を削除します。

[詳細はこちら](#)

- ZREMRANGEBYLEX

ソート対象集合の辞書的範囲内のメンバーを削除します。すべてのメンバーを削除した場合、ソート対象集合を削除します。

[詳細はこちら](#)

- ZREMRANGEBYRANK

ソート対象集合のインデックス範囲内のメンバーを削除します。すべてのメンバーを削除した場合、ソート対象集合を削除します。

[詳細はこちら](#)

- ZREMRANGEBYSCORE

スコアの範囲内にあるソート対象集合のメンバーを削除します。すべてのメンバーを削除した場合、ソート対象集合を削除します。

[詳細はこちら](#)

- ZREVRANGE

ソート対象集合のインデックス範囲内のメンバーを逆の順序で返します。

[詳細はこちら](#)

- ZREVRANGEBYLEX

ソート対象集合の辞書的範囲内のメンバーを逆の順序で返します。

[詳細はこちら](#)

- ZREVRANGEBYSCORE

ソート対象集合のスコア範囲内のメンバーを逆の順序で返します。

[詳細はこちら](#)

- ZREVRANK

ソート対象集合内のメンバーのインデックスをスコアの降順で返します。

[詳細はこちら](#)

- ZSCAN

ソート対象集合のメンバーとスコアを反復処理します。

[詳細はこちら](#)

- ZSCORE

ソート対象集合内のメンバーのスコアを返します。

[詳細はこちら](#)

- ZUNION

複数のソート対象集合の和を返します。

[詳細はこちら](#)

- ZUNIONSTORE

複数のソート対象集合の和をキーに格納します。

[詳細はこちら](#)

ストリームコマンド

- XACK

ストリームのコンシューマーグループメンバーによって正常に承認されたメッセージの数を返します。

[詳細はこちら](#)

- XADD

ストリームに新しいメッセージを追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- XAUTOCLAIM

メッセージがコンシューマーグループメンバーに配信されたかのように、コンシューマーグループ内のメッセージの所有権を変更または取得します。

[詳細はこちら](#)

- XCLAIM

メッセージがコンシューマーグループメンバーに配信されたかのように、コンシューマーグループ内のメッセージの所有権を変更または取得します。

[詳細はこちら](#)

- XDEL

ストリームからメッセージを削除し、メッセージ数を返します。

[詳細はこちら](#)

- XGROUP CREATE

コンシューマーグループを作成します。

[詳細はこちら](#)

- XGROUP CREATECONSUMER

コンシューマーグループにコンシューマーを作成します。

[詳細はこちら](#)

- XGROUP DELCONSUMER

コンシューマーグループからコンシューマーを削除します。

[詳細はこちら](#)

- XGROUP DESTROY

コンシューマーグループを破棄します。

[詳細はこちら](#)

- XGROUP SETID

コンシューマーグループの最後に配信された ID を設定します。

[詳細はこちら](#)

- XINFO CONSUMERS

コンシューマーグループ内のコンシューマーのリストを返します。

[詳細はこちら](#)

- XINFO GROUPS

ストリームのコンシューマーグループのリストを返します。

[詳細はこちら](#)

- XINFO STREAM

ディスクに関する情報を返します。

[詳細はこちら](#)

- XLEN

ストリーム内のメッセージ数を返します。

[詳細はこちら](#)

- XPENDING

ストリームコンシューマーグループの保留中のエントリリストから情報とエントリを返します。

[詳細はこちら](#)

- XRANGE

ID の範囲内のストリームからのメッセージを返します。

[詳細はこちら](#)

- XREAD

複数のストリームから、リクエストされた ID よりも大きい ID のメッセージを返します。メッセージが使用可能になるまでブロックします。

[詳細はこちら](#)

- XREADGROUP

グループ内のコンシューマのストリームから新規または過去のメッセージを返します。メッセージが使用可能になるまでブロックします。

[詳細はこちら](#)

- XREVRANGE

ID の範囲内のストリームからのメッセージを逆の順序で返します。

[詳細はこちら](#)

- XTRIM

ストリームの先頭からメッセージを削除します。

[詳細はこちら](#)

文字列コマンド

- APPEND

キーの値に文字列を追加します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- DECR

キーの整数値を 1 ずつ減らします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- DECRBY

キーの整数値から数値を減らします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- GET

キーの文字列値を返します。

[詳細はこちら](#)

- GETDEL

キーを削除した後にキーの文字列値を返します。

[詳細はこちら](#)

- GETEX

キーの有効期限を設定した後にキーの文字列値を返します。

[詳細はこちら](#)

- GETRANGE

キーに格納されている文字列の部分文字列を返します。

[詳細はこちら](#)

- GETSET

キーを新しい値に設定した後に、そのキーの以前の文字列値を返します。

[詳細はこちら](#)

- INCR

キーの整数値を 1 ずつ増やします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- INCRBY

キーの整数値を数値分増やします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- INCRBYFLOAT

ハッシュ内のキーの浮動小数点値を数値分増やします。フィールドが存在しない場合、0 を初期値として使用します。

[詳細はこちら](#)

- LCS

最も長い共通部分文字列を検索します。

[詳細はこちら](#)

- MGET

1 つまたは複数のキーの文字列値をアトミックに返します。

[詳細はこちら](#)

- MSET

1 つまたは複数のキーの文字列値をアトミックに作成または変更します。

[詳細はこちら](#)

- MSETNX

1 つまたは複数のキーについて、どのキーも存在しない場合にのみ、その文字列値をアトミックに変更します。

[詳細はこちら](#)

- PSETEX

キーの文字列値とミリ秒単位の有効期限の両方を設定します。キーが存在しない場合はキーが作成されます。

[詳細はこちら](#)

- SET

タイプを無視して、キーの文字列値を設定します。キーが存在しない場合はキーが作成されます。

[詳細はこちら](#)

- SETEX

キーの文字列値と有効期限を設定します。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- SETNX

キーが存在しない場合にのみ、キーの文字列値を設定します。

[詳細はこちら](#)

- SETRANGE

文字列値の一部をオフセットにより別の文字列で上書きします。キーが存在しない場合は、キーを作成します。

[詳細はこちら](#)

- STRLEN

文字列値の長さを返します。

[詳細はこちら](#)

- SUBSTR

文字列値から部分文字列を返します。

[詳細はこちら](#)

トランザクションコマンド

- DISCARD

トランザクションを破棄します。

[詳細はこちら](#)

- EXEC

トランザクション内のすべてのコマンドを実行します。

[詳細はこちら](#)

- MULTI

トランザクションをスタートします。

[詳細はこちら](#)

制限される Redis コマンド

マネージドサービス操作性を実現するため、ElastiCache では高度な特権を必要とする特定のキャッシュエンジン固有のコマンドへのアクセスが制限されます。Redis を実行するキャッシュの場合、以下のコマンドは使用できません。

- `acl setuser`
- `acl load`
- `acl save`
- `acl deluser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster addslotsrange`
- `cluster bumpepoch`
- `cluster delslot`
- `cluster delslotsrange`
- `cluster failover`
- `cluster flushslots`
- `cluster forget`
- `cluster links`
- `cluster meet`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `psync`
- `replicaof`
- `save`
- `slaveof`
- `shutdown`
- `sync`

また、以下のコマンドはサーバーレスキャッシュでは使用できません。

- `acl log`

- `client caching`
- `client getreidir`
- `client id`
- `client info`
- `client kill`
- `client list`
- `client no-evict`
- `client pause`
- `client tracking`
- `client trackinginfo`
- `client unblock`
- `client unpause`
- `cluster count-failure-reports`
- `fcall`
- `fcall_ro`
- `function`
- `function delete`
- `function dump`
- `function flush`
- `function help`
- `function kill`
- `function list`
- `function load`
- `function restore`
- `function stats`
- `keys`
- `lastsave`
- `latency`
- `latency doctor`
- `latency graph`

- latency help
- latency histogram
- latency history
- latency latest
- latency reset
- memory
- memory doctor
- memory help
- memory malloc-stats
- memory purge
- memory stats
- memory usage
- monitor
- move
- object
- object encoding
- object freq
- object help
- object idletime
- object refcount
- pfdebug
- pfselftest
- psubscribe
- pubsub numpat
- punsubscribe
- script kill
- slowlog
- slowlog get
- slowlog help
- slowlog len

- `slowlog reset`
- `swapdb`
- `unwatch`
- `wait`
- `watch`

Redis の設定と制限事項

Redis エンジンには多数の設定パラメータがあり、その中には ElastiCache for Redis で変更可能なものもあれば、安定したパフォーマンスと信頼性を提供するために変更できないものもあります。

サーバーレスキャッシュ

サーバーレスキャッシュでは、パラメータグループは使用されず、変更できる Redis 設定はありません。次の Redis パラメータが設定されています。

名前	詳細	説明
<code>acl-pubsub-default</code>	<code>allchannels</code>	キャッシュ上の ACL ユーザーの、デフォルトの Pubsub チャンネルのアクセス許可。
<code>client-output-buffer-limit</code>	<code>normal 0 0 0</code> <code>pubsub 32mb 8mb 60</code>	通常のクライアントにはバッファ制限はありません。PUB/SUB クライアントが 32 MiB のバックログに違反したり、8MiB のバックログに 60 秒間違反したりすると、接続が切断されます。
<code>client-query-buffer-limit</code>	1 GiB	単一のクライアントクエリバッファのサイズ上限。また、クライアントは 4,000 個を超える引数を持つクエストを発行することはできません。
<code>cluster-allow-pubsubshard-when-down</code>	<code>yes</code>	これにより、キャッシュが部分的にダウンしていても、キャッシュは Pubsub トラフィックを処理できます。

名前	詳細	説明
<code>cluster-allow-reads-when-down</code>	yes	これにより、キャッシュが部分的にダウンしていても、キャッシュは read トラフィックを処理できます。
<code>cluster-enabled</code>	yes	サーバーレスキャッシュはすべてクラスターモードが有効になっているため、データを複数のバックエンドシャードに透過的にパーティション化できます。すべてのスロットは、単一の仮想ノードが所有するものとしてクライアントに表示されます。
<code>cluster-require-full-coverage</code>	no	キースペースが部分的にダウンした場合 (つまり、少なくとも 1 つのハッシュスロットにアクセスできなくなった場合)、キャッシュはキースペースのうちまだカバーされている部分のクエリを受け付け続けます。キースペース全体は常に <code>cluster slots</code> 内の 1 つの仮想ノードによって「カバー」されます。
<code>lua-time-limit</code>	5000	<p>ElastiCache がスクリプトを停止するアクションを実行までの Lua スクリプトの最大実行時間 (ミリ秒単位)。</p> <p><code>lua-time-limit</code> を超過した場合、すべての Redis コマンドによりエラーが <code>___-BUSY</code> の形式で返されます。この状態により、多く必須 Redis 操作との干渉が発生する可能性があるため、ElastiCache はまず <code>SCRIPT KILL</code> コマンドを発行します。これに失敗すると、ElastiCache は強制的に Redis を再開します。</p>
<code>maxclients</code>	65000	キャッシュに一度に接続できるクライアントの最大数。それ以上接続を確立すると、成功することもあるが、失敗することもあります。

名前	詳細	説明
maxmemory-policy	volatile-lru	TTL が設定されているアイテムは、キャッシュのメモリ制限に達すると、least-recently-used (LRU) の推定に従って削除されます。
notify-keyspace-events	(空の文字列)	キースペースイベントは現在、サーバーレスキャッシュではサポートされていません。
port	プライマリポート: 6379 読み取りポート: 6380	サーバーレスキャッシュは、同じホスト名の 2 つのポートをアドバタイズします。プライマリポートは書き込みと読み取りを許可し、読み取りポートは READONLY コマンドを使用して低レイテンシーで最終的に一貫性のある読み取りを可能にします。
proto-max-bulk-len	512 MiB	1 つの要素リクエストのサイズ上限。
timeout	0	クライアントは特定のアイドル時に強制的に切断されることはありませんが、負荷分散のために定常状態では切断される場合があります。

さらに、次の制限があります。

名前	詳細	説明
キー名の長さ	4 KiB	単一の Redis キーまたはチャンネル名のサイズ上限。クライアントがこれよりも大きいキーを参照すると、エラーが発生します。
Lua スクリプトのサイズ	4 MiB	単一の Redis Lua スクリプトのサイズ上限。これよりも大きい Lua スクリプトを読み込もうとすると、エラーが発生します。

名前	詳細	説明
スロットサイズ	32 GiB	単一の Redis ハッシュスロットのサイズ上限。クライアントが 1 つの Redis スロットにこれよりも多くのデータを設定しようとする、そのスロットのエビクションポリシーがトリガーされ、削除可能なキーがない場合はメモリ不足 (OOM) エラーが表示されます。

独自設計型クラスター

独自設計型クラスターの場合、設定パラメータのデフォルト値と設定可能な値については、「[Redis 固有のパラメータ](#)」を参照してください。オーバーライドが必要な特定のユースケースがない限り、通常はデフォルト値を推奨します。

Redis クライアントに関するベストプラクティス

一般的なシナリオのベストプラクティスと、定評のあるオープンソース Redis クライアントライブラリ (redis-py、PHPRedis、Lettuce) のコード例を紹介します。

トピック

- [多数の接続](#)
- [Redis クラスターのクライアント検出とエクスポネンシャルバックオフ](#)
- [クライアント側のタイムアウトを設定する](#)
- [サーバー側のアイドルタイムアウトの設定](#)
- [Redis Lua スクリプト](#)
- [大きな複合アイテムの保管](#)
- [Lettuce クライアント設定](#)
- [IPv6 クライアントの例](#)

多数の接続

サーバーレスキャッシュと個々の ElastiCache for Redis ノードは、最大 65,000 の同時クライアント接続をサポートしています。ただし、パフォーマンスを最適化するために、クライアントアプリケーションが常にはそのレベルの接続で動作しないことをお勧めします。Redis は、クライアントの受信

リクエストが順番に処理されるイベントループに基づくシングルスレッドのプロセスです。つまり、接続しているクライアントの数が増えると、特定のクライアントの応答時間が長くなります。

Redis サーバーで接続のボトルネックを回避するために、以下の対策を実行します。

- リードレプリカからの読み取り操作を実行する。これは、クラスターモードを無効にした状態で ElastiCache リーダーエンドポイントを使用するか、サーバーレスキャッシュを含め、クラスターモードを有効にした状態で読み取り用のレプリカを使用することで実行できます。
- 書き込みトラフィックを複数のプライマリノードに分散する。これには 2 つの方法があります。マルチシャード Redis クラスターは、Redis クラスターモード対応のクライアントで使用できます。また、クライアント側のシャーディングで無効になっているクラスターモードで、複数のプライマリノードに書き込むこともできます。これはサーバーレスキャッシュで自動的に行われます。
- クライアントライブラリで利用可能な場合は、接続プールを使用する。

一般に、TCP 接続の作成は、一般的な Redis コマンドに比べて計算コストが高いオペレーションです。例えば、既存の接続を再利用すると、SET/GET リクエストの処理が桁違いに速くなります。有限サイズのクライアント接続プールを使用すると、接続管理のオーバーヘッドが軽減されます。また、クライアントアプリケーションからの同時着信接続数も制限されます。

次の PHPRedis のコード例は、新しいユーザーのリクエストごとに新しい接続が作成されることを示しています。

```
$redis = new Redis();
if ($redis->connect($HOST, $PORT) != TRUE) {
    //ERROR: connection failed
    return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

このコードは、Graviton2 (m6g.2xlarge) ElastiCache for Redis ノードに接続された Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上のループでベンチマークされています。クライアントとサーバーは同じアベイラビリティーゾーンに配置されています。オペレーション全体の平均レイテンシーは 2.82 ミリ秒でした。

コードを更新して永続的な接続と接続プールを使用した場合、オペレーション全体の平均レイテンシーは 0.21 ミリ秒でした。

```
$redis = new Redis();
if ($redis->pconnect($HOST, $PORT) != TRUE) {
    // ERROR: connection failed
    return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

必要な redis.ini の設定:

- redis.pconnect.pooling_enabled=1
- redis.pconnect.connection_limit=10

以下のコードは [Redis-py 接続プール](#) の例です。

```
conn = Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
    max_connections=10))
conn.set(key, value)
```

以下のコードは [Lettuce 接続プール](#) の例です。

```
RedisClient client = RedisClient.create(RedisURI.create(HOST, PORT));
GenericObjectPool<StatefulRedisConnection> pool =
    ConnectionPoolSupport.createGenericObjectPool(() -> client.connect(), new
    GenericObjectPoolConfig());
pool.setMaxTotal(10); // Configure max connections to 10
try (StatefulRedisConnection connection = pool.borrowObject()) {
    RedisCommands syncCommands = connection.sync();
    syncCommands.set(key, value);
}
```

Redis クラスターのクライアント検出とエクスポネンシャルバックオフ

クラスターモードを有効にして ElastiCache for Redis クラスターに接続する場合、該当する Redis クライアントライブラリがクラスター対応であることが必要です。クライアントは、適切なノードにリクエストを送信し、クラスターのリダイレクト処理によるパフォーマンスのオーバーヘッドを回避できるように、ハッシュスロットとクラスター内のノードを対応付けたマップを取得する必要があります。そのため、クライアントは以下の 2 つの異なる状況下で、スロットとマッピング先のノードを網羅したリストを検出する必要があります。

- クライアントが初期化され、初期スロット構成を読み込む必要がある。
- MOVED リダイレクトをサーバーから受信した。例えば、フェイルオーバー時に元のプライマリノードのスロットをすべてレプリカに引き継ぐ場合や、リシャーディング時にスロットがソースプライマリノードからターゲットプライマリノードに移動される場合が該当します。

クライアント検出は通常、Redis サーバーに CLUSTER SLOT コマンドまたは CLUSTER NODE コマンドを発行することで行われます。CLUSTER SLOT メソッドを推奨します。その理由は、一連のスロット範囲と、関連するプライマリノードとレプリカノードをクライアントに返すからです。その場合、クライアント側で別途解析を行う必要がなく、効率が向上します。

クラスタポートロジによっては、CLUSTER SLOT コマンドの応答のサイズがクラスタのサイズに応じて変わる場合があります。ノード数が多い大きなクラスタは、応答も大きくなります。したがって、クラスタポートロジ検出を行うクライアントの数が、際限なく増えないようにすることが重要です。例えば、クライアントアプリケーションの起動時やサーバーとの接続の切断時にクラスタ検出を実行しなければならない場合に、クライアントアプリケーションで再接続や検出のリクエストを複数回行い、再試行時のエクスポネンシャルバックオフが実装されていないという間違いがよく見受けられます。そのせいで CPU 使用率が 100% になり、そのまま Redis サーバーが長時間応答しなくなる可能性があります。各 CLUSTER SLOT コマンドでクラスタバス内の多数のノードを処理しなければならない場合は、こうした停止状態が長引きます。このような動作が原因でクライアントが停止する事態は、これまでも、Python (redis-py-cluster) や Java (Lettuce と Redisson) などのさまざまな言語でいくつも確認されています。

サーバーレスキャッシュでは、アドバタイズされるクラスタポートロジが静的であり、書き込みエンドポイントと読み取りエンドポイントの 2 つのエントリで構成されるため、こうした問題の多くは自動的に軽減されます。また、キャッシュエンドポイントを使用する場合、クラスタ検出が自動的に複数のノードに分散されます。ただし、以下の推奨事項は引き続き有効です。

接続リクエストや検出リクエストが殺到した場合の影響を軽減するために、以下の対応を推奨します。

- クライアントアプリケーションからの同時着信接続数を制限するために、有限サイズのクライアント接続プールを実装する。
- タイムアウトによりクライアントがサーバーから切断された場合は、エクスポネンシャルバックオフとジッター(揺らぎ)を加えて再試行する。これにより、複数のクライアントが同時にサーバーに負荷をかける事態を阻止できます。
- 「[接続エンドポイントの検索](#)」のガイドを参考にして、クラスタエンドポイントを検索し、クラスタ検出を実行する。これにより、クラスタ内のハードコーディングされたいくつかのシード

ノードにアクセスする代わりに、検出の負荷をクラスター内のすべてのノード (最大 90 個) 間で分散できます。

redis-py、PHPRedis、Lettuce におけるエクスポネンシャルバックオフの再試行ロジックのコード例を以下に紹介します。

バックオフロジックのサンプル 1: redis-py

redis-py には、障害の発生直後に 1 回再試行する再試行メカニズムが組み込まれています。このメカニズムは、[Redis](#) オブジェクトの作成時に渡される `retry_on_timeout` 引数で有効にすることができます。ここでは、エクスポネンシャルバックオフとジッターを加えたカスタムの再試行メカニズムを紹介します。[redis-py \(#1494\)](#) で、エクスポネンシャルバックオフをネイティブに実装するプルリクエストを送信しました。今後は、手動で実装する必要がなくなる可能性があります。

```
def run_with_backoff(function, retries=5):
    base_backoff = 0.1 # base 100ms backoff
    max_backoff = 10 # sleep for maximum 10 seconds
    tries = 0
    while True:
        try:
            return function()
        except (ConnectionError, TimeoutError):
            if tries >= retries:
                raise
            backoff = min(max_backoff, base_backoff * (pow(2, tries) + random.random()))
            print(f"sleeping for {backoff:.2f}s")
            sleep(backoff)
            tries += 1
```

その後、以下のコードを使用して値を設定できます。

```
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
    max_connections=10))
res = run_with_backoff(lambda: client.set("key", "value"))
print(res)
```

ワークロードに応じて、例えば、レイテンシーの影響を受けやすいワークロードについては、バックオフの基準値を 1 秒から数十ミリ秒または数百ミリ秒に変更することができます。

バックオフロジックのサンプル 2: PHPRedis

PHPRedis には、最大 10 回 (回数設定は不可) 再試行する再試行メカニズムが組み込まれています。試行間隔の遅延を設定できます (2 回目以降にジッターを加えます)。詳細については、[こちらのサンプルコード](#)を参照してください。[PHPRedis \(#1986\)](#) で、エクスポネンシャルバックオフをネイティブに実装するプルリクエストを送信しました。このリクエストはその後統合され、[文書化](#)されています。PHPRedis の最新リリースを使用している場合は、手動で実装する必要はありませんが、以前のバージョンを使用している場合の参考資料として、ここで紹介しておきます。差し当たり、再試行メカニズムの遅延を設定するコード例を以下に紹介します。

```
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, $timeout, NULL, $retry_interval) != TRUE) {
    return; // ERROR: connection failed
}
$client->set($key, $value);
```

バックオフロジックのサンプル 3: Lettuce

Lettuce には、[エクスポネンシャルバックオフとジッター](#)の投稿で説明したエクスポネンシャルバックオフ戦略に基づく再試行メカニズムが組み込まれています。以下は、フルジッターのアプローチを示すコードの抜粋です。

```
public static void main(String[] args)
{
    ClientResources resources = null;
    RedisClient client = null;

    try {
        resources = DefaultClientResources.builder()
            .reconnectDelay(Delay.fullJitter(
                Duration.ofMillis(100), // minimum 100 millisecond delay
                Duration.ofSeconds(5), // maximum 5 second delay
                100, TimeUnit.MILLISECONDS) // 100 millisecond base
            ).build();

        client = RedisClient.create(resources, RedisURI.create(HOST, PORT));
        client.setOptions(ClientOptions.builder()
            .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
            100 millisecond connection timeout
            .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(5)).build()) //
            5 second command timeout
            .build());
```

```
    // use the connection pool from above example
} finally {
    if (connection != null) {
        connection.close();
    }

    if (client != null){
        client.shutdown();
    }

    if (resources != null){
        resources.shutdown();
    }

}
}
```

クライアント側のタイムアウトを設定する

サーバーがリクエストを処理してレスポンスを生成するのに十分な時間を確保できるように、クライアント側のタイムアウトを適切に設定します。また、これにより、サーバーへの接続を確立できない場合でも、フェイルファストが可能です。Redis コマンドの中には、他のコマンドよりも計算コストが高いものがあります。例えば、アトミックに実行する必要がある複数のコマンドを含む Lua スクリプトや MULTI/EXEC トランザクションなどです。一般的には、以下を含むサーバーからレスポンスを受け取る前にクライアントがタイムアウトするのを避けるため、クライアント側のタイムアウト時間を長くすることが推奨されます。

- 複数のキーにまたがるコマンドの実行
- 複数の個別の Redis コマンドで構成される MULTI/EXEC トランザクションまたは Lua スクリプトの実行
- 大きな値の読み取り
- BLPOP などのブロッキング操作の実行

BLPOP のようなブロッキング操作の場合、ベストプラクティスとして、コマンドのタイムアウトをソケットのタイムアウトよりも小さい数値に設定します。

redis-py、PHPRedis、および Lettuce でクライアント側のタイムアウトを実装するコード例を以下に示します。

タイムアウトの設定例 1: redis-py

redis-py を使用したコード例は次のとおりです。

```
# connect to Redis server with a 100 millisecond timeout
# give every Redis command a 2 second timeout
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
    max_connections=10,socket_connect_timeout=0.1,socket_timeout=2))

res = client.set("key", "value") # will timeout after 2 seconds
print(res)                       # if there is a connection error

res = client.blpop("list", timeout=1) # will timeout after 1 second
                                     # less than the 2 second socket timeout
print(res)
```

タイムアウトの設定例 2: PHPRedis

PHPRedis を使用したコード例は次のとおりです。

```
// connect to Redis server with a 100ms timeout
// give every Redis command a 2s timeout
$client = new Redis();
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, 0.1, NULL, 100, $read_timeout=2) != TRUE){
    return; // ERROR: connection failed
}
$client->set($key, $value);

$res = $client->set("key", "value"); // will timeout after 2 seconds
print "$res\n";                    // if there is a connection error

$res = $client->blpop("list", 1); // will timeout after 1 second
print "$res\n";                    // less than the 2 second socket timeout
```

タイムアウトの設定例 3: Lettuce

Lettuce を使用したコード例は次のとおりです。

```
// connect to Redis server and give every command a 2 second timeout
```

```
public static void main(String[] args)
{
    RedisClient client = null;
    StatefulRedisConnection<String, String> connection = null;
    try {
        client = RedisClient.create(RedisURI.create(HOST, PORT));
        client.setOptions(ClientOptions.builder()
            .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
            100 millisecond connection timeout
            .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(2)).build()) //
            2 second command timeout
            .build());

        // use the connection pool from above example

        commands.set("key", "value"); // will timeout after 2 seconds
        commands.blpop(1, "list"); // BLPPOP with 1 second timeout
    } finally {
        if (connection != null) {
            connection.close();
        }

        if (client != null){
            client.shutdown();
        }
    }
}
```

サーバー側のアイドルタイムアウトの設定

お客様のアプリケーションにアイドル状態のクライアントが多数接続されていて、コマンドを活発に送信しているわけではないケースが散見されています。このような場合、多数のアイドル状態のクライアントで 65,000 の接続数を使い果たしてしまう可能性があります。こうした状況を回避するには、[Redis 固有のパラメータ](#) を使用してサーバーのタイムアウトを適切に設定してください。そうすれば、サーバーがアイドル状態のクライアントの接続を積極的に切断するため、接続数の上昇を防ぐことができます。この設定は、サーバーレスキャッシュでは使用できません。

Redis Lua スクリプト

Redis は 200 超のコマンドをサポートしており、その中には、Lua スクリプトを実行するコマンドもあります。ただし、Lua スクリプトに関しては、Redis のメモリと可用性に影響しかねない注意点がいくつかあります。

パラメータ化されていない Lua スクリプト

各 Lua スクリプトは、実行前に Redis サーバーにキャッシュされます。パラメータ化されていない Lua スクリプトはそれぞれが異なるため、Redis サーバーで大量の Lua スクリプトが保存され、メモリ消費量が増える可能性があります。これを軽減するには、すべての Lua スクリプトを確実にパラメータ化し、SCRIPT FLUSH を定期的に実行して、キャッシュされている Lua スクリプトを適宜クリーンアップします。

次のコード例では、パラメータ化したスクリプトの使い方を紹介します。まず、パラメータ化しない場合の例を紹介します。この場合、3 つの異なる Lua スクリプトがキャッシュされるため、推奨されません。

```
eval "return redis.call('set','key1','1')" 0
eval "return redis.call('set','key2','2')" 0
eval "return redis.call('set','key3','3')" 0
```

代わりに、以下のパターンを使用して、渡されたパラメータを受け入れることができる単一のスクリプトを作成してください。

```
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key1 1
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key2 2
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key3 3
```

実行時間の長い Lua スクリプト

Lua スクリプトは複数のコマンドをアトミックに実行できるため、通常の Redis コマンドよりも所要時間が長くなる場合があります。Lua スクリプトが読み取り専用のオペレーションのみを実行する場合は、途中で停止できます。ただし、Lua スクリプトが書き込みオペレーションを実行した時点で強制終了できなくなり、最後まで実行しなければなりません。変更処理を行う Lua スクリプトの実行時間が長引くと、Redis サーバーが長時間応答しなくなる可能性があります。この問題を軽減するには、実行時間の長い Lua スクリプトを避け、実稼働前の環境でスクリプトをテストしてください。

ステルス書き込みを行う Lua スクリプト

Redis が maxmemory を上回っても、Lua スクリプトが引き続き Redis に新しいデータを書き込むケースがいくつかあります。

- Redis サーバーが maxmemory を下回っている場合にスクリプトが開始し、そのスクリプト内に複数の書き込みオペレーションが含まれている。

- スクリプトの最初の書き込みコマンドはメモリを消費しないが (DEL など)、後続の複数の書き込みオペレーションがメモリを消費する。
- Redis サーバーで noeviction 以外の適切なエビクションポリシーを設定することで、この問題を軽減できます。これにより、Redis は Lua スクリプトの合間にアイテムを削除し、メモリを解放できます。

大きな複合アイテムの保管

状況によっては、アプリケーションが大きな複合アイテム (マルチ GB のハッシュデータセットなど) を Redis に保存することがあります。これは、Redis でパフォーマンスの問題が生じやすくなるため、原則としては推奨されません。例えば、クライアントは HGETALL コマンドを実行して、マルチ GB のハッシュコレクション全体を取得できます。この場合、クライアントの出力バッファに大きなアイテムがバッファリングされ、Redis サーバーに多大なメモリ負荷がかかる可能性があります。また、クラスターモードでのスロット移行では、ElastiCache はシリアル化されたサイズが 256 MB を超えるアイテムを含むスロットを移行しません。

大きいアイテムの問題を解決するために、以下の点を推奨します。

- 大きな複合アイテムは複数の小さなアイテムに分割する。例えば、大きなハッシュコレクションを、そのコレクションを適切に反映したキー名スキームを使用して (アイテムのコレクションを識別する共通のプレフィックスをキー名に付けるなど)、個々のキーと値のフィールドに分割します。同じコレクション内の複数のフィールドにアトミックにアクセスする必要がある場合は、MGET コマンドを使用して、同一コマンドで複数のキーと値のペアを取得できます。
- どの方法を検討しても大きなコレクションデータセットを分割できない場合は、コレクション全体ではなく、コレクション内のデータのサブセットを操作するコマンドを使用してみる。マルチ GB のコレクション全体を同一コマンドでアトミックに取得しなければならないようなユースケースは避けてください。ハッシュコレクションに対して HGETALL の代わりに HGET コマンドや HMGET コマンドを使用することが、その一例です。

Lettuce クライアント設定

このセクションでは、推奨される Java と Lettuce の設定オプションと、それらを ElastiCache クラスターに適用する方法について説明します。

このセクションの推奨事項は、Lettuce バージョン 6.2.2 でテスト済みです。

トピック

- [例: クラスターモードで TLS が有効な場合の Lettuce の設定](#)
- [例: クラスターモードが無効で TLS が有効な場合の Lettuce の設定](#)

Java DNS キャッシュ TTL

Java 仮想マシン (JVM) は DNS 名参照をキャッシュします。JVM がホスト名を IP アドレスに変換するとき、time-to-live (TTL) と呼ばれる指定期間 IP アドレスをキャッシュします。

TTL 値の選択は、レイテンシーおよび変化に対する応答性と間のトレードオフです。TTL を短くすると、DNS リゾルバーはクラスターの DNS の更新をより早く認識します。これにより、クラスターで実行される置換やその他のワークフローにアプリケーションがより迅速に応答できるようになります。ただし、TTL が低すぎると、クエリの量が増え、それによってアプリケーションのレイテンシーが増加する可能性があります。絶対的に正しい TTL 値は存在しませんが、TTL 値を設定するときは、変更が有効になるまで待つことができる時間の長さについて検討する必要があります。

ElastiCache ノードは、変更される可能性がある DNS 名を使用するため、5 秒 ~ 10 秒の低い TTL 値で JVM を設定することをお勧めします。これにより、ノードの IP アドレスが変更されたときに、アプリケーションは DNS エントリに対して再度クエリを実行することで、リソースの新しい IP アドレスを取得し、使用できるようになります。

一部の Java 設定では JVM のデフォルトの TTL が設定されるため、JVM が再起動されるまで、DNS エントリが更新されることはありません。

JVM TTL を設定する方法の詳細については、「[JVM TTL を設定する方法](#)」を参照してください。

Lettuce のバージョン

Lettuce のバージョン 6.2.2 以降の使用をお勧めします。

エンドポイント

クラスターモードが有効なクラスターを使用している場合は、redisUri をクラスター設定エンドポイントに設定します。この URI の DNS ルックアップは、クラスターで使用可能なすべてのノードのリストを返し、クラスターの初期化中にそれらのノードの 1 つにランダムに解決されます。トポロジ更新の仕組みの詳細については、このトピックで後述する「DynamicRefreshResources」を参照してください。

SocketOption

[KeepAlive](#) を有効にします。このオプションを有効にすると、コマンドのランタイムに失敗した接続を処理する必要が減ります。

[接続タイムアウト](#)は、アプリケーションの要件とワークロードに基づいて設定してください。詳細については、このトピックで後述する「タイムアウト」のセクションを参照してください。

ClusterClientOption: クラスターモードが有効なクライアントオプション

接続が失われたときは [AutoReconnect](#) を有効にします。

[CommandTimeout](#) を設定します。詳細については、このトピックで後述する「タイムアウト」のセクションを参照してください。

[NodeFilter](#) を設定すると、障害が発生したノードをトポロジから除外できます。Lettuce は、「クラスターノード」出力にあるすべてのノード (PFAIL/FAIL ステータスのノードを含む) をクライアントの「パーティション」 (シャードとも呼ばれます) に保存します。クラスタートポロジを作成するプロセスで、すべてのパーティションノードに接続を試みます。障害が発生したノードを追加する Lettuce の動作は、何らかの理由でノードが交換されるときに接続エラー (または警告) を引き起こす可能性があります。

例えば、フェイルオーバーが完了してクラスターが回復プロセスを開始した後、clusterTopology が更新されている間、ダウンしているノードがトポロジから完全に削除されるまで、クラスターバスノードマップにそれが FAIL ノードとして短期間リストされます。この間、Lettuce Redis クライアントはそのノードを正常なノードと見なし、継続的に接続します。そのため、再試行を使い果たすとエラーが発生します。

例:

```
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
        ... // other options
        .nodeFilter(it ->
            ! (it.is(RedisClusterNode.NodeFlag.FAIL)
                || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
                || it.is(RedisClusterNode.NodeFlag.HANDSHAKE)
                || it.is(RedisClusterNode.NodeFlag.NOADDR)))
        .validateClusterNodeMembership(false)
        .build();
redisClusterClient.setOptions(clusterClientOptions);
```

Note

ノードフィルタリングは、DynamicRefreshSources を true に設定して使用するのが最適です。そうしないと、1つの問題のあるシードノードからトポロジビューを取得すると、一部

のシャードのプライマリノードに障害が発生していると見なされ、このプライマリノードは除外され、スロットがカバーされなくなります。(DynamicRefreshSources が true の場合に) 複数のシードノードが存在すると、この問題が発生する可能性が低くなります。これは、新しく昇格したプライマリとのフェイルオーバー後に、少なくとも一部のシードノードでトポロジビューを更新する必要があるためです。

ClusterTopologyRefreshOptions: クラスターモード対応クライアントのクラスタートポロジ更新を制御するオプション

Note

クラスターモードが無効なクラスターは、クラスター検出コマンドをサポートしていないため、すべてのクライアントの動的トポロジ検出機能と互換性があるわけではありません。

ElastiCache で無効になっているクラスターモードは Lettuce の

MasterSlaveTopologyRefresh と互換性がありません。代わりに、クラスターモードが無効になっている場合は、StaticMasterReplicaTopologyProvider を設定し、クラスターの読み取りと書き込みのエンドポイントを提供します。

クラスターモードが無効なクラスターとの接続の詳細については、「[Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)」を参照してください。

Lettuce の動的トポロジ検出機能を使いたい場合は、既存のクラスターと同じシャード構成でクラスターモードが有効なクラスターを作成できます。ただし、クラスターモードが有効なクラスターでは、高速フェールオーバーをサポートするために、少なくとも 3 つのシャードと 1 つのレプリカを構成することをお勧めします。

[enablePeriodicRefresh](#) を有効にします。これにより、クラスタートポロジを定期的に更新できるため、クライアントは refreshPeriod の間隔 (デフォルト:60 秒) でクラスタートポロジを更新できます。無効にすると、クライアントはクラスターに対してコマンドの実行を試みたときにエラーが発生した場合にのみ、クラスタートポロジを更新します。

このオプションを有効にすると、このジョブをバックグラウンドタスクに追加することで、クラスタートポロジの更新に伴うレイテンシを減らすことができます。トポロジの更新はバックグラウンドジョブで実行されますが、多数のノードがあるクラスターでは多少遅くなる可能性があります。これは、すべてのノードが最新のクラスタービューを取得するためにそれらのビューに対してクエリが実行されているためです。大規模なクラスターを実行する場合は、この時間を長くすることをお勧めします。

[enableAllAdaptiveRefreshTriggers](#) を有効にします。これにより、MOVED_REDIRECT、ASK_REDIRECT、PERSISTENT_RECONNECTS、UNCOVERED_SLOT、UNK のすべての [トリガー](#) を使用する適応型トポロジ更新が可能になります。適応型更新トリガーは、Redis クラスター操作中に発生したイベントに基づいてトポロジビューの更新を開始します。このオプションを有効にすると、前述のトリガーのいずれかが発生すると、トポロジがすぐに更新されます。適応型更新トリガーは、イベントが大規模で発生する可能性があるため (更新間のデフォルトタイムアウトは 30)、タイムアウトを使用してレート制限されます。

[closeStaleConnections](#) を有効にします。これにより、クラスタートポロジを更新するときに、古い接続を閉じることができます。[ClusterTopologyRefreshOptions.isPeriodicRefreshEnabled\(\)](#) が true の場合にのみ有効になります。有効にすると、クライアントは古い接続を閉じて新しい接続をバックグラウンドで作成できます。これにより、コマンドのランタイムに失敗した接続を処理する必要が減ります。

[dynamicRefreshResources](#) を有効にします。小規模なクラスターでは `DynamicRefreshResources` を有効にし、大規模なクラスターでは無効にすることをお勧めします。`DynamicRefreshResources` を使用すると、提供されたシードノード (クラスター構成エンドポイントなど) からクラスターノードを検出できます。検出されたすべてのノードを、クラスタートポロジを更新するためのソースとして使用します。

動的更新を使用すると、検出されたすべてのノードにクラスタートポロジを照会し、最も正確なクラスタービューを選択しようと試みます。false に設定すると、最初のシードノードのみがトポロジ検出のソースとして使用され、クライアント数は最初のシードノードについてのみ取得されます。無効になっている場合、クラスター設定エンドポイントが障害の発生したノードに解決されたとき、クラスタービューを更新しようとする失敗し、例外が発生します。このシナリオは、障害が発生したノードのエントリがクラスター設定エンドポイントから削除されるまでに時間がかかるときに発生する可能性があります。そのため、設定エンドポイントは、障害が発生したノードに短期間ランダムに解決できます。

ただし、有効にすると、クラスタービューから受信したすべてのクラスターノードを使用して、現在のビューについてクエリを実行します。障害が発生したノードをそのビューから除外するので、トポロジ更新は成功します。ただし、`dynamicRefreshSources` が true の場合、Lettuce はすべてのノードにクエリを実行してクラスタービューを取得し、結果を比較します。そのため、多数のノードを持つクラスターではコストがかかる可能性があります。多数のノードがあるクラスターでは、この機能をオフにすることをお勧めします。

```
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
```

```
.enablePeriodicRefresh()  
.dynamicRefreshSources(true)  
.build();
```

ClientResources

[DnsResolver](#) を [DirContextDnsResolver](#) で設定します。DNS リゾルバーは Java の `com.sun.jndi.dns.DnsContextFactory` をベースにしています。

[reconnectDelay](#) をエクスポネンシャルバックオフとフルジッターで設定します。Lettuce には、エクスポネンシャルバックオフ戦略に基づく再試行メカニズムが組み込まれています。詳細については、AWS アーキテクチャブログの「[エクスポネンシャルバックオフとジッター](#)」を参照してください。再試行バックオフ戦略の重要性に関する詳細については、AWS データベースブログの[ベストプラクティスブログ投稿](#)のバックオフロジックのセクションを参照してください。

```
ClientResources clientResources = DefaultClientResources.builder()  
    .dnsResolver(new DirContextDnsResolver())  
    .reconnectDelay(  
        Delay.fullJitter(  
            Duration.ofMillis(100),    // minimum 100 millisecond delay  
            Duration.ofSeconds(10),    // maximum 10 second delay  
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base  
    .build();
```

Timeouts

コマンドのタイムアウトよりも低い接続タイムアウト値を使用してください。Lettuce はレイジー接続確立を使用します。そのため、接続タイムアウトがコマンドタイムアウトよりも大きい場合、Lettuce が異常なノードへの接続を試みてコマンドのタイムアウトが常に超過すると、トポロジ更新後に障害が一定期間持続する可能性があります。

異なるコマンドに対しては動的コマンドタイムアウトを使用してください。コマンドの想定期間に基づいてコマンドタイムアウトを設定することをお勧めします。例えば、FLUSHDB、FLUSHALL、KEYS、SMEMBERS、Lua スクリプトなど、複数のキーを反復処理するコマンドにはタイムアウトを長く設定します。SET、GET、HSET など、1 つのキーコマンドではタイムアウトを短くします。

Note

次の例で設定されているタイムアウトは、最大 20 バイトの長さのキーと値で SET/GET コマンドを実行したテスト用です。コマンドが複雑な場合や、キーと値が大きい場合は、処理時

間が長くなる可能性があります。タイムアウトは、アプリケーションのユースケースに基づいて設定する必要があります。

```
private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);
```

```
SocketOptions socketOptions = SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .build();
```

```
class DynamicClusterTimeout extends TimeoutSource {
    private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
    ImmutableSet.<ProtocolKeyword>builder()
        .add(CommandType.FLUSHDB)
        .add(CommandType.FLUSHALL)
        .add(CommandType.CLUSTER)
        .add(CommandType.INFO)
        .add(CommandType.KEYS)
        .build();

    private final Duration defaultCommandTimeout;
    private final Duration metaCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

// Use a dynamic timeout for commands, to avoid timeouts during
```

```
// cluster management and slow operations.  
TimeoutOptions timeoutOptions = TimeoutOptions.builder()  
.timeoutSource(  
    new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))  
.build();
```

例: クラスターモードで TLS が有効な場合の Lettuce の設定

Note

次の例のタイムアウトは、最大 20 バイトの長さのキーと値で SET/GET コマンドを実行したテスト用です。コマンドが複雑な場合や、キーと値が大きい場合は、処理時間が長くなる可能性があります。タイムアウトは、アプリケーションのユースケースに基づいて設定する必要があります。

```
// Set DNS cache TTL  
public void setJVMProperties() {  
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");  
}  
  
private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);  
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);  
// Socket connect timeout should be lower than command timeout for Lettuce  
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);  
  
// Create RedisURI from the cluster configuration endpoint  
clusterConfigurationEndpoint = <cluster-configuration-endpoint> // TODO: add your  
    cluster configuration endpoint  
final RedisURI redisUriCluster =  
    RedisURI.Builder.redis(clusterConfigurationEndpoint)  
        .withPort(6379)  
        .withSsl(true)  
        .build();  
  
// Configure the client's resources  
ClientResources clientResources = DefaultClientResources.builder()  
    .reconnectDelay(  
        Delay.fullJitter(  
            Duration.ofMillis(100), // minimum 100 millisecond delay  
            Duration.ofSeconds(10), // maximum 10 second delay  
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
```

```
.dnsResolver(new DirContextDnsResolver())
    .build();

// Create a cluster client instance with the URI and resources
RedisClusterClient redisClusterClient =
    RedisClusterClient.create(clientResources, redisUriCluster);

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
class DynamicClusterTimeout extends TimeoutSource {
    private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
        ImmutableSet.<ProtocolKeyword>builder()
            .add(CommandType.FLUSHDB)
            .add(CommandType.FLUSHALL)
            .add(CommandType.CLUSTER)
            .add(CommandType.INFO)
            .add(CommandType.KEYS)
            .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT,
        META_COMMAND_TIMEOUT))
    .build();

// Configure the topology refreshment options
final ClusterTopologyRefreshOptions topologyOptions =
```

```
ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();

// Configure the socket options
final SocketOptions socketOptions =
    SocketOptions.builder()
        .connectTimeout(CONNECT_TIMEOUT)
        .keepAlive(true)
        .build();

// Configure the client's options
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
        .topologyRefreshOptions(topologyOptions)
        .socketOptions(socketOptions)
        .autoReconnect(true)
        .timeoutOptions(timeoutOptions)
        .nodeFilter(it ->
            ! (it.is(RedisClusterNode.NodeFlag.FAIL)
                || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
                || it.is(RedisClusterNode.NodeFlag.NOADDR)))
        .validateClusterNodeMembership(false)
        .build();

redisClusterClient.setOptions(clusterClientOptions);

// Get a connection
final StatefulRedisClusterConnection<String, String> connection =
    redisClusterClient.connect();

// Get cluster sync/async commands
RedisAdvancedClusterCommands<String, String> sync = connection.sync();
RedisAdvancedClusterAsyncCommands<String, String> async = connection.async();
```

例: クラスターモードが無効で TLS が有効な場合の Lettuce の設定

Note

次の例のタイムアウトは、最大 20 バイトの長さのキーと値で SET/GET コマンドを実行したテスト用です。コマンドが複雑な場合や、キーと値が大きい場合は、処理時間が長くなる可

能性があります。タイムアウトは、アプリケーションのユースケースに基づいて設定する必要があります。

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the primary/reader endpoint
clusterEndpoint = <primary/reader-endpoint> // TODO: add your node endpoint
RedisURI redisUriStandalone =

    RedisURI.Builder.redis(clusterEndpoint).withPort(6379).withSsl(true).withDatabase(0).build();

ClientResources clientResources =
    DefaultClientResources.builder()
        .dnsResolver(new DirContextDnsResolver())
        .reconnectDelay(
            Delay.fullJitter(
                Duration.ofMillis(100), // minimum 100 millisecond delay
                Duration.ofSeconds(10), // maximum 10 second delay
                100,
                TimeUnit.MILLISECONDS)) // 100 millisecond base
        .build();

// Use a dynamic timeout for commands, to avoid timeouts during
// slow operations.
class DynamicTimeout extends TimeoutSource {
    private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
    ImmutableSet.<ProtocolKeyword>builder()
        .add(CommandType.FLUSHDB)
        .add(CommandType.FLUSHALL)
        .add(CommandType.INFO)
        .add(CommandType.KEYS)
        .build();

    private final Duration metaCommandTimeout;
```

```
private final Duration defaultCommandTimeout;

DynamicTimeout(Duration defaultTimeout, Duration metaTimeout)
{
    defaultCommandTimeout = defaultTimeout;
    metaCommandTimeout = metaTimeout;
}

@Override
public long getTimeout(RedisCommand<?, ?, ?> command) {
    if (META_COMMAND_TYPES.contains(command.getType())) {
        return metaCommandTimeout.toMillis();
    }
    return defaultCommandTimeout.toMillis();
}
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
    .build();

final SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(CONNECT_TIMEOUT).keepAlive(true).build();

ClientOptions clientOptions =

    ClientOptions.builder().timeoutOptions(timeoutOptions).socketOptions(socketOptions).build();

RedisClient redisClient = RedisClient.create(clientResources, redisUriStandalone);
redisClient.setOptions(clientOptions);
```

IPv6 クライアントの例

一般的に使用されているオープンソースのクライアントライブラリで IPv6 ElastiCache 対応リソースを操作する場合のベストプラクティスを以下に示します。 [ElastiCacheリソース用のクライアントの設定に関する推奨事項については、対話に関する既存のベストプラクティスを参照できます。](#) ElastiCache ただし、IPv6 対応リソースを操作する際には、注意すべき点がいくつかあります。

検証済みクライアント

ElastiCache オープンソースの Redis と互換性があります。つまり、IPv6 接続をサポートするオープンソース Redis クライアントは、Redis クラスターで IPv6 が有効になっている ElastiCache IPv6 に接続できるはずですが、さらに、最も一般的な Python および Java クライアントのいくつかは、サ

ポートされているすべてのネットワークタイプ設定 (IPv4 のみ、IPv6 のみ、およびデュアルスタック) で動作するように特別にテストおよび検証されています。

検証済みクライアント:

- [Redis Py \(\) - 4.1.2](#)
- [Lettuce - バージョン: 6.1.6.リリース](#)
- [Jedis - バージョン: 3.6.0](#)

デュアルスタッククラスターの優先プロトコルの設定

クラスターモードが有効な Redis クラスターでは、IP 検出パラメータを使用して、クライアントがクラスター内のノードに接続するために使用するプロトコルを制御できます。IP 検出パラメータは、IPv4 または IPv6 に設定できます。

Redis クラスターの場合、IP 検出パラメータは、[クラスタースロット \(\)](#)、[クラスターシャード \(\)](#)、[クラスターノード \(\)](#) の出力で使用される IP プロトコルを設定します。これらのコマンドは、クライアントがクラスタートポロジを検出するために使用されます。クライアントは、これらのコマンドの IP を使用して、クラスター内の他のノードに接続します。

IP 検出を変更しても、接続しているクライアントのダウンタイムは発生しません。ただし、変更が反映されるまで時間がかかる場合があります。Redis クラスターに変更が完全に反映されたかどうかを判断するには、cluster slots の出力をモニタリングします。cluster slots コマンドによって返されたすべてのノードが新しいプロトコルの IP を報告すると、変更の反映が完了します。

Redis-Py を使った例:

```
cluster = RedisCluster(host="xxxx", port=6379)
target_type = IPv6Address # Or IPv4Address if changing to IPv4

nodes = set()
while len(nodes) == 0 or not all((type(ip_address(host)) is target_type) for host in
    nodes):
    nodes = set()

    # This refreshes the cluster topology and will discovery any node updates.
    # Under the hood it calls cluster slots
    cluster.nodes_manager.initialize()
    for node in cluster.get_nodes():
        nodes.add(node.host)
    self.logger.info(nodes)
```

```
time.sleep(1)
```

Lettuce の例:

```
RedisClusterClient clusterClient = RedisClusterClient.create(RedisURI.create("xxxx",
6379));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
switching to IPv4

Set<String> nodes;

do {
    // Check for any changes in the cluster topology.
    // Under the hood this calls cluster slots
    clusterClient.refreshPartitions();
    Set<String> nodes = new HashSet<>();

    for (RedisClusterNode node : clusterClient.getPartitions().getPartitions()) {
        nodes.add(node.getUri().getHost());
    }

    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
    try {
        return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
    } catch (UnknownHostException ignored) {}
    return false;
})));
```

TLS 対応のデュアルスタッククラスター ElastiCache

TLS がクラスターで有効になっている場合、ElastiCache クラスター検出機能 (cluster slots、cluster shards、cluster nodes) は IP の代わりにホスト名を返します。その後、IP ElastiCache の代わりにホスト名を使用してクラスターに接続し、TLS ハンドシェイクを実行します。つまり、クライアントは IP 検出パラメータの影響を受けません。TLS が有効なクラスターでは、IP 検出パラメータは優先 IP プロトコルに影響しません。代わりに、使用する IP プロトコルは、DNS ホスト名を解決する際にクライアントがどの IP プロトコルを使用するかによって決まります。

Java クライアント

IPv4 と IPv6 の両方をサポートする Java 環境から接続する場合、後方互換性のために Java はデフォルトで IPv6 よりも IPv4 を優先します。ただし、IP プロトコルプリファレンスは JVM 引数を使用して設定できます。IPv4 を優先するには、JVM は `-Djava.net.preferIPv4Stack=true` を受け入れ、IPv6 セット `-Djava.net.preferIPv6Stack=true` を優先します。 `-Djava.net.preferIPv4Stack=true` を設定すると、JVM は IPv6 接続を行わなくなります。これらを他の Redis 以外のアプリケーションにも含めます。

ホストレベルの設定

一般に、クライアントまたはクライアントランタイムに IP プロトコルプリファレンスを設定するための構成オプションが提供されていない場合、DNS 解決を実行するとき、IP プロトコルはホストの設定に依存します。デフォルトでは、ほとんどのホストは IPv4 よりも IPv6 を優先しますが、この優先度はホストレベルで設定できます。これは、クラスターへの DNS リクエストだけでなく、そのホストからの DNS リクエストすべてに影響します。ElastiCache

Linux ホスト

Linux では、`gai.conf` ファイルを変更して IP プロトコルプリファレンスを設定できます。 `gai.conf` ファイルは `/etc/gai.conf` の下にあります。 `gai.conf` の指定がない場合は、 `/usr/share/doc/glibc-common-x.xx/gai.conf` に `/etc/gai.conf` にコピーできるサンプルを用意し、デフォルトの設定をコメント解除する必要があります。ElastiCache クラスターへの接続時に IPv4 を優先するように構成を更新するには、クラスター IP を含む CIDR 範囲の優先順位をデフォルトの IPv6 接続の優先順位よりも高くするように更新します。デフォルトでは、IPv6 接続の優先順位は 40 です。例えば、クラスターが CIDR `172.31.0.0/16` のサブネットにあると仮定すると、以下の設定では、クライアントはそのクラスターへの IPv4 接続を優先することになります。

```
label ::1/128      0
label ::/0         1
label 2002::/16   2
label ::/96        3
label ::ffff:0:0/96 4
label fec0::/10   5
label fc00::/7    6
label 2001:0::/32  7
label ::ffff:172.31.0.0/112 8
#
# This default differs from the tables given in RFC 3484 by handling
# (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
# The reason for this difference is that these addresses are never
# NATed while IPv4 site-local addresses most probably are. Given
# the precedence of IPv6 over IPv4 (see below) on machines having only
```

```
# site-local IPv4 and IPv6 addresses a lookup for a global address would
# see the IPv6 be preferred. The result is a long delay because the
# site-local IPv6 addresses cannot be used while the IPv4 address is
# (at least for the foreseeable future) NATed. We also treat Teredo
# tunnels special.
#
# precedence <mask> <value>
# Add another rule to the RFC 3484 precedence table. See section 2.1
# and 10.3 in RFC 3484. The default is:
#
precedence ::1/128          50
precedence ::/0            40
precedence 2002::/16       30
precedence ::/96           20
precedence ::ffff:0:0/96   10
precedence ::ffff:172.31.0.0/112 100
```

gai.conf の詳細については、[Linux のメインページ](#)を参照してください。

Windows ホスト

Windows ホストのプロセスも同様です。Windows ホストの場合は netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL を実行できます。これは Linux ホストで gai.conf ファイルを変更するのと同じ効果があります。

これにより、指定された CIDR 範囲の IPv6 接続よりも IPv4 接続を優先するように優先設定ポリシーが更新されます。例えば、クラスターが 172.31.0.0/16 CIDR のサブネット内にあると仮定した場合、netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15 を実行すると、次の優先順位表になり、クライアントがクラスターに接続する際に IPv4 を優先することになります。

```
C:\Users\Administrator>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence Label Prefix
-----
100 15 ::ffff:172.31.0.0:0/112
20 4 ::ffff:0:0/96
50 0 ::1/128
40 1 ::/0
30 2 2002::/16
5 5 2001::/32
```

```
3 13 fc00::/7
1 11 fec0::/10
1 12 3ffe::/16
1 3 ::/96
```

予約メモリの管理

予約メモリは、nondata 用に確保されるメモリです。バックアップまたはフェイルオーバーを実行すると、Redis は、クラスターのデータが .rdb ファイルに書き込まれる間にクラスターに書き込みオペレーションを記録している、使用可能なメモリを使用します。すべての書き込みに十分なメモリが使用可能できない場合、プロセスは失敗します。以下に、ElastiCache for Redis の予約メモリを管理するためのオプションと、それらのオプションを適用する方法について説明します。

トピック

- [予約メモリはどれくらい必要ですか。](#)
- [予約メモリを管理するパラメータ](#)
- [予約メモリ管理パラメータの指定](#)

予約メモリはどれくらい必要ですか。

がバックアップとレプリケーションプロセス ElastiCache を実装する方法が異なるため、経験則は reserved-memory-percent パラメータを使用してノードタイプの maxmemory 値の 25% を予約することです。これはデフォルト値であり、ほとんどの場合に推奨されます。

バースト可能なマイクロインスタンスタイプとスモールインスタンスタイプが maxmemory 制限近くで動作している場合、スワップの使用が発生する可能性があります。バックアップ、レプリケーション、高トラフィック時にこれらのインスタンスタイプの運用上の信頼性を向上させるには、reserved-memory-percent パラメータの値を小さなインスタンスタイプでは最大 30%、マイクロインスタンスタイプでは最大 50% に増やすことをお勧めします。

データ階層化を使用する ElastiCache クラスターの書き込みが多いワークロードでは、ノードの使用可能なメモリの 50% reserved-memory-percent までを増やすことをお勧めします。

詳細については、次を参照してください。

- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)
- [同期とバックアップの実装方法](#)
- [データ階層化](#)

予約メモリを管理するパラメータ

2017年3月16日現在、Amazon ElastiCache for Redisには、Redisメモリを管理するための2つの相互に排他的なパラメータ `reserved-memory` と `reserved-memory-percent` が用意されています。これらのパラメータのいずれも Redis ディストリビューションには含まれていません。

ElastiCache 顧客になった時期に応じて、これらのパラメータの1つまたはもう1つがデフォルトのメモリ管理パラメータです。このパラメータは、新しい Redis クラスターまたはレプリケーショングループを作成し、デフォルトのパラメータグループを使用する場合に適用されます。

- 2017年3月16日より前に開始したユーザーでは — デフォルトのパラメータグループを使用して Redis クラスターまたはレプリケーショングループを作成する場合、メモリ管理パラメータは `reserved-memory` になります。この場合、0バイトのメモリが予約されます。
- 2017年3月16日以降に開始したユーザーでは — デフォルトのパラメータグループを使用して Redis クラスターまたはレプリケーショングループを作成する場合、メモリ管理パラメータは `reserved-memory-percent` になります。この場合、ノードの `maxmemory` 値の25%がデータ以外の目的で予約されます。

2つの Redis メモリ管理パラメータの説明を確認したならば、デフォルトではないものかデフォルト以外の値のものを使用することもできます。その場合は、他の予約メモリ管理パラメータに変更できます。

そのパラメータの値を変更するには、カスタムパラメータグループを作成し、希望のメモリ管理パラメータと値を使用するように変更します。これで、新しい Redis クラスターまたはレプリケーショングループを作成するたびに、そのカスタムパラメータグループを使用できるようになります。既存のクラスターまたはレプリケーショングループの場合は、カスタムパラメータグループを使用するように変更できます。

詳細については、次を参照してください。

- [予約メモリ管理パラメータの指定](#)
- [パラメータグループを作成する](#)
- [パラメータグループを変更する](#)
- [ElastiCache クラスターの変更](#)
- [レプリケーショングループの変更](#)

予約メモリのパラメータ

2017年3月16日より前は、Redisの予約済みメモリ管理 ElastiCache 用はすべてパラメータを使用して行われていました reserved-memory。reserved-memory のデフォルト値は 0 です。このデフォルトは Redis のオーバーヘッド用にメモリを確保せず、Redis はノードのメモリすべてをデータ用に消費できます。

バックアップ用およびフェイルオーバー用に使用できる十分なメモリを持てるように reserved-memory を変更するには、カスタムパラメータグループを作成する必要があります。このカスタムパラメータグループで、reserved-memory をクラスターおよびクラスターのノードタイプで実行している Redis のバージョンに適切な値に設定します。詳細については、「[予約メモリはどれくらい必要ですか。](#)」を参照してください。

ElastiCache for Redis パラメータ reserved-memory は for Redis に ElastiCache 固有であり、Redis ディストリビューションの一部ではありません。

次の手順で reserved-memory を使用して Redis クラスターのメモリを管理する方法を示します。

予約メモリを使用してメモリを予約するには

1. 実行中のエンジンバージョンに一致するパラメータグループファミリーを指定するカスタムパラメータグループを作成します。たとえば、redis2.8 パラメータグループファミリーを指定します。詳細については、「[パラメータグループを作成する](#)」を参照してください。

```
aws elasticache create-cache-parameter-group \  
  --cache-parameter-group-name redis6x-m3x1 \  
  --description "Redis 2.8.x for m3.xlarge node type" \  
  --cache-parameter-group-family redis6.x
```

2. Redis のオーバーヘッドのために予約するメモリのバイト数を計算します。ノードタイプに対する maxmemory 値を [Redis のノードタイプ固有のパラメータ](#) で確認できます。
3. パラメータ reserved-memory が前の手順で計算したバイト数であるように、カスタムパラメータグループを変更します。次の AWS CLI 例では、2.8.22 より前のバージョンの Redis を実行し、ノードの の半分を予約する必要があることを前提としています maxmemory。詳細については、「[パラメータグループを変更する](#)」を参照してください。

```
aws elasticache modify-cache-parameter-group \  
  --cache-parameter-group-name redis28-m3x1 \  
  --parameter-name-values "ParameterName=reserved-memory,  
  ParameterValue=7130316800"
```

各ノードタイプには異なる maxmemory 値があるため、使用する各ノードタイプに対して個別のカスタムパラメータグループが必要です。したがって、各ノードタイプには reserved-memory に対して異なる値が必要です。

4. カスタムパラメータグループを使用するように Redis クラスターまたはレプリケーショングループを変更します。

次の CLI の例では、カスタムパラメータグループ redis28-m3x1 をすぐに使用するようにクラスター my-redis-cluster を変更しています。詳細については、「[ElastiCache クラスターの変更](#)」を参照してください。

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-redis-cluster \  
  --cache-parameter-group-name redis28-m3x1 \  
  --apply-immediately
```

次の CLI の例では、カスタムパラメータグループ redis28-m3x1 をすぐに使用するようにレプリケーショングループ my-redis-repl-grp を変更しています。詳細については、「[レプリケーショングループの変更](#)」。

```
aws elasticache modify-replication-group \  
  --replication-group-id my-redis-repl-grp \  
  --cache-parameter-group-name redis28-m3x1 \  
  --apply-immediately
```

reserved-memory-percent パラメータ

2017 年 3 月 16 日、Amazon はパラメータ ElastiCache を導入 reserved-memory-percent し、ElastiCache for Redis のすべてのバージョンで利用可能にしました。reserved-memory-percent の目的は、すべてのクラスターに対して予約メモリ管理を簡易化することです。ノードタイプにかかわらずクラスターの予約メモリを管理するために、各パラメータグループファミリー (redis2.8 など) に対して単一のパラメータグループを持てるようにすることによって実行します。reserved-memory-percent のデフォルト値は 25 (25 パーセント) です。

ElastiCache for Redis パラメータ reserved-memory-percent は ElastiCache for Redis に固有であり、Redis ディストリビューションの一部ではありません。

r6gd ファミリーのノードを使用しているクラスターでメモリ使用量が 75% に達すると、データ階層化が自動的にトリガーされます。詳細については、「[データ階層化](#)」を参照してください。

を使用してメモリを予約するには `reserved-memory-percent`

`reserved-memory-percent` を使用して ElastiCache for Redis クラスターのメモリを管理するには、次のいずれかを実行します。

- Redis 2.8.22 以降を実行している場合は、クラスターに、デフォルトのパラメータグループを割り当てます。デフォルトの 25 パーセントで十分です。そうでない場合、次のステップを実行して、値を変更します。
- 2.8.22 より前の Redis のバージョンを実行している場合、おそらく `reserved-memory-percent` のデフォルトの 25 パーセントよりも多くのメモリを確保する必要があります。そのため、次の手順を使用します。

のパーセント値を変更するには `reserved-memory-percent`

1. 実行中のエンジンバージョンに一致するパラメータグループファミリーを指定するカスタムパラメータグループを作成します。たとえば、`redis2.8` パラメータグループファミリーを指定します。カスタムパラメータグループは、デフォルトのパラメータグループを変更できないため必要です。詳細については、「[パラメータグループを作成する](#)」を参照してください。

```
aws elasticache create-cache-parameter-group \  
  --cache-parameter-group-name redis28-50 \  
  --description "Redis 2.8.x 50% reserved" \  
  --cache-parameter-group-family redis2.8
```

`reserved-memory-percent` は、ノードの `maxmemory` に対する割合としてメモリを予約するため、各ノードタイプに対応するカスタムパラメータグループは必要ありません。

2. `reserved-memory-percent` が 50 (50 パーセント) であるようにカスタムパラメータグループを変更します。詳細については、「[パラメータグループを変更する](#)」を参照してください。

```
aws elasticache modify-cache-parameter-group \  
  --cache-parameter-group-name redis28-50 \  
  --parameter-name-values "ParameterName=reserved-memory-percent,  
  ParameterValue=50"
```

3. 2.8.22 より前の Redis のバージョンを実行している Redis クラスターまたはレプリケーショングループに対して、このカスタムパラメータグループを使用します。

次の CLI の例では、カスタムパラメータグループ `redis28-50` をすぐに使用するように Redis クラスター `my-redis-cluster` を変更しています。詳細については、「[ElastiCache クラスターの変更](#)」を参照してください。

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-redis-cluster \  
  --cache-parameter-group-name redis28-50 \  
  --apply-immediately
```

次の CLI の例では、カスタムパラメータグループ `redis28-50` をすぐに使用するように Redis レプリケーショングループ `my-redis-repl-grp` を変更しています。詳細については、「[レプリケーショングループの変更](#)」を参照してください。

```
aws elasticache modify-replication-group \  
  --replication-group-id my-redis-repl-grp \  
  --cache-parameter-group-name redis28-50 \  
  --apply-immediately
```

予約メモリ管理パラメータの指定

2017 年 3 月 16 日に現在の ElastiCache 顧客であった場合、デフォルトのリザーブメモリ管理パラメータ `reserved-memory` は 0 (0) バイトのリザーブメモリです。2017 年 3 月 16 日以降に ElastiCache 顧客になった場合、デフォルトのリザーブ `reserved-memory-percent` ドメモリ管理パラメータはノードのメモリの 25% が予約されています。これは、ElastiCache for Redis クラスターまたはレプリケーショングループをいつ作成したかに関係なく当てはまります。ただし、リザーブメモリ管理パラメータは、AWS CLI または ElastiCache API を使用して変更できます。

パラメータ `reserved-memory` および `reserved-memory-percent` は相互に排他的です。パラメータグループには、常にどちらかがありますが、両方があることはありません。パラメータグループを変更することによって、パラメータグループが予約メモリ管理のためにどちらのパラメータを使用するかを変更できます。デフォルトのパラメータグループは変更できないため、パラメータグループはカスタムパラメータグループである必要があります。詳細については、「[パラメータグループを作成する](#)」を参照してください。

を指定するには `reserved-memory-percent`

予約メモリ管理パラメータとして `reserved-memory-percent` を使用するには、`modify-cache-parameter-group` コマンドを使用してカスタムパラメータグループを変更しま

す。parameter-name-values パラメータを使用して、reserved-memory-percent とその値を指定します。

次の CLI の例では、reserved-memory-percent を使用して予約メモリを管理するように、カスタムパラメータグループ redis32-cluster-on を変更します。パラメータグループが予約メモリ管理に ParameterName パラメータを使用するには、ParameterValue に値を割り当てる必要があります。詳細については、「[パラメータグループを変更する](#)」を参照してください。

```
aws elasticache modify-cache-parameter-group \  
  --cache-parameter-group-name redis32-cluster-on \  
  --parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=25"
```

reserved-memory を指定するには

予約メモリ管理パラメータとして reserved-memory を使用するには、modify-cache-parameter-group コマンドを使用してカスタムパラメータグループを変更します。parameter-name-values パラメータを使用して、reserved-memory とその値を指定します。

次の CLI の例では、reserved-memory を使用して予約メモリを管理するように、カスタムパラメータグループ redis32-m3x1 を変更します。パラメータグループが予約メモリ管理に ParameterName パラメータを使用するには、ParameterValue に値を割り当てる必要があります。エンジンバージョンは 2.8.22 より新しいため、値を cache.m3.xlarge の maxmemory の 25% である 3565158400 に設定します。詳細については、「[パラメータグループを変更する](#)」を参照してください。

```
aws elasticache modify-cache-parameter-group \  
  --cache-parameter-group-name redis32-m3x1 \  
  --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```

独自設計型クラスターを使用する場合のベストプラクティス

このセクションは、独自の Redis クラスターを設計する場合にのみ適用されます。以下のベストプラクティスを確認し、参考にすることをお勧めします。

トピック

- [マルチ AZ によるダウンタイムの最小化](#)
- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)
- [オンラインクラスターのサイズ変更](#)
- [メンテナンス中のダウンタイムを最小限に抑える](#)

マルチ AZ によるダウンタイムの最小化

[マルチ AZ とダウンタイムの最小化の詳細については、「マルチ AZ による Redis のダウンタイムの最小化 ElastiCache」](#)を参照してください。

Redis スナップショットを作成するのに十分なメモリがあることの確認

バージョン 2.8.22 以降の Redis のスナップショットと同期

Redis 2.8.22 で分岐なしの保存プロセスが導入されました。これにより、同期および保存中にスワップを使用することなく、アプリケーションにより多くのメモリを割り当てて使用することができます。詳細については、「[同期とバックアップの実装方法](#)」を参照してください。

バージョン 2.8.22 以前の Redis のスナップショットおよび同期

Redis を操作すると ElastiCache、Redis は次のような場合にバックグラウンド書き込みコマンドを呼び出します。

- バックアップのためのスナップショットを作成するとき。
- レプリカとレプリケーショングループ内のプライマリを同期させるとき。
- Redis の AOF (Append-Only File) 機能を有効にするとき。
- レプリカをプライマリに昇格するとき (プライマリ/レプリカの同期が実行される)。

Redis がバックグラウンドの書き込みプロセスを実行するときは、常に、このプロセスのオーバーヘッドに対応するのに十分なメモリが利用できる必要があります。十分なメモリを利用できない場合、このプロセスは失敗します。このため、Redis クラスターの作成時には、十分なメモリがあるノードインスタンスタイプを選択することが重要です。

バックグラウンド書き込みプロセスとメモリ使用率

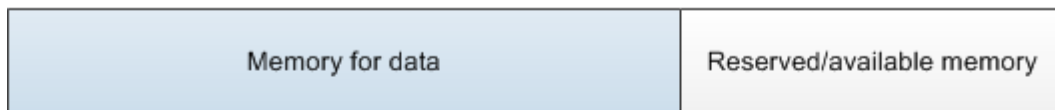
バックグラウンド書き込みプロセスが呼び出されると、Redis は常にそのプロセスを生成 (フォーク) します (Redis はシングルスレッドであることを思い出してください)。1 つのフォークがデータをディスクの Redis .rdb スナップショットファイルに永続化します。もう 1 つのフォークは、すべての読み取りと書き込みのオペレーションを処理します。スナップショットが point-in-time スナップショットであることを確認するために、すべてのデータ更新と追加は、データ領域とは別の使用可能なメモリ領域に書き込まれます。

データをディスクに保持しながら、すべての書き込みオペレーションを記録するのに十分なメモリが利用できる限り、メモリ不足の問題は発生しません。次のいずれかに該当する場合は、メモリ不足の問題が発生する可能性があります。

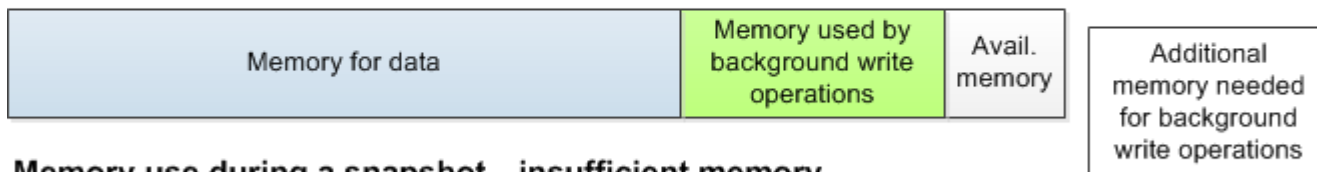
- アプリケーションで頻繁に書き込みオペレーションが実行され、新しいデータや更新されたデータを受け入れるために使用可能なメモリが大量に必要なになる。
- 新しいデータや更新されたデータを書き込むために使用できるメモリが少なすぎる。
- ディスクに永続化するのに長時間かかる大規模なデータセットがあり、大量の書き込みオペレーションが必要になる。

次の図は、バックグラウンド書き込みプロセス実行時のメモリの使用を示しています。

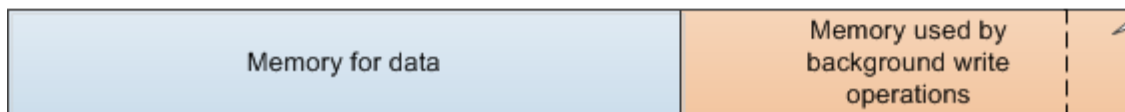
Memory use prior to a snapshot



Memory use during a snapshot—sufficient memory



Memory use during a snapshot—insufficient memory



バックアップを実行する際のパフォーマンスへの影響については、「[独自設計型クラスターのバックアップがパフォーマンスに与える影響](#)」を参照してください。

Redis がスナップショットを実行する方法の詳細については、<http://redis.io> を参照してください。

リージョンとアベイラビリティゾーンの詳細については、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

バックグラウンド書き込み実行中のメモリ不足の回避

BGSAVE または BGREWRITEAOF のようなバックグラウンド書き込みプロセスが呼び出されると、プロセスの失敗を防ぐためには、処理中の書き込みオペレーションが消費する量以上のメモリが必要となります。最悪の場合のシナリオでは、バックグラウンド書き込みのオペレーション中にすべての Redis レコードが更新され、新しいレコードがキャッシュに追加されます。そのため、Redis バージョン 2.8.22 より前の場合は、reserved-memory-percent を 50 (50 パーセント) に設定し、Redis バージョン 2.8.22 以降の場合は 25 (25 パーセント) に設定することをお勧めします。

maxmemory 値は、データとオペレーションのオーバーヘッドで使用できるメモリを示します。デフォルトのパラメータグループの reserved-memory パラメータを変更することはできないため、

クラスター用のカスタムパラメータグループを作成する必要があります。reserved-memory のデフォルト値は 0 です。この場合、Redis はすべての maxmemory をデータ用に消費でき、バックグラウンド書き込みプロセスなどの他の用途に使用できるメモリがほとんど残されない可能性があります。ノードインスタンスタイプごとの maxmemory 値については、「[Redis のノードタイプ固有のパラメータ](#)」を参照してください。

ボックスで reserved-memory パラメータを使用して、Redis のメモリ使用量を抑えることができます。

の Redis 固有のパラメータの詳細については ElastiCache、「」を参照してください [Redis 固有のパラメータ](#)。

パラメータグループの作成と変更については、「[パラメータグループを作成する](#)」と「[パラメータグループを変更する](#)」を参照してください。

オンラインクラスターのサイズ変更

リシャーディングには、クラスターへのシャードまたはノードの追加と削除、およびキースペースの再分散が含まれます。したがって、クラスターの負荷、メモリ使用率、データ全体のサイズなど、シャーディングオペレーションには複数のものが影響します。最適なエクスペリエンスを得るには、均一なワークロードパターンディストリビューションのクラスターベストプラクティス全体に従うことをお勧めします。さらに、次のステップを実行することをお勧めします。

リシャーディングを開始する前に、次のことをお勧めします:

- アプリケーションをテストする – 可能であれば、ステージング環境でリシャーディング中にアプリケーションの動作をテストします。
- [スケーリング問題の早期通知の取得] – リシャーディングは計算処理能力を集中的に使用するオペレーションです。このため、リシャーディング中は CPU 使用率をマルチコアインスタンスで 80% 未満、シングルコアインスタンスで 50% 未満にすることをお勧めします。ElastiCache for Redis メトリックスをモニタリングして、アプリケーションでスケーリングの問題が発生する前にリシャーディングを開始します。追跡すると有用なメトリックスは、CPUUtilization、NetworkBytesIn、NetworkBytesOut、CurrConnections、NewConnections です。
- スケーリングする前に、空きメモリが十分に確保されていることを確認する – スケーリングする場合、保持するシャードの空きメモリが、削除するシャードに使用されているメモリの 1.5 倍以上であることを確認します。
- オフピーク時にリシャーディングを開始する – このプラクティスは、リシャーディングオペレーションがクライアントのレイテンシーとスループットに与える影響を軽減するのに役立ちます。ま

た、スロット再分散に多くのリソースを使用できるため、リシャーディングをより迅速に完了できます。

- クライアントのタイムアウト動作を確認する – オンラインクラスターのサイズ変更中に、一部のクライアントでレイテンシーが長くなる場合があります。より大きなタイムアウトでクライアントライブラリを設定すると、サーバーがより高い負荷条件でもシステムが接続する時間を与えることができます。場合によっては、サーバーへの接続を多数開く必要があります。この場合、エクスポンENTIALバックオフを追加してロジックを再接続することを検討してください。こうすると、サーバーに対して大量の新しい接続が同時に行われるのを防ぐことができます。
- すべてのシャードに関数を読み込む – クラスターをスケールアウトすると、ElastiCache は (ランダムに選択された) 既存のノードのいずれかにロードされた関数を新しいノードに自動的にレプリケートします。クラスターに Redis 7.0 以上があり、アプリケーションで [Redis 関数](#) を使用している場合は、スケールアウトする前に、クラスターがシャードにより異なる関数定義にならないように、すべての関数をすべてのシャードに読み込むことをお勧めします。

リシャーディング後は、以下の点に注意してください:

- ターゲットのシャードで十分なメモリが利用できない場合、スケールインが部分的に成功している可能性があります。そのような結果が生じた場合、必要に応じて使用可能なメモリーを確認し、オペレーションを再試行してください。ターゲットのシャードのデータは削除されません。
- 大きなアイテムのスロットは移行されません。特に、シリアル化後に 256 MB を超えるアイテムを持つスロットは移行されません。
- FLUSHALL および FLUSHDB コマンドは、リシャーディング操作中の Lua スクリプト内ではサポートされません。Redis 6 より前では、移行中の BRPOPLPUSH スロットで動作するコマンドはサポートされていません。

メンテナンス中のダウンタイムを最小限に抑える

クラスターモード設定を使用して、マネージド型またはアンマネージド型のオペレーション中に可用性を最大限に高めることができます。クラスター検出エンドポイントに接続するクラスターモードがサポートされるクライアントを使用することをお勧めします。クラスターモードを無効にした場合は、すべての書き込みオペレーションにプライマリエンドポイントを使用することをお勧めします。

読み取りアクティビティの場合、アプリケーションはクラスター内のいずれのノードにも接続できます。プライマリエンドポイントとは異なり、ノードエンドポイントは特定のエンドポイントに解決されます。レプリカの追加または削除など、クラスターに変更を加えた場合は、アプリケーション

でノードエンドポイントを更新する必要があります。このため、クラスターモードを無効にする場合は、読み取りアクティビティにリーダーエンドポイントを使用することをお勧めします。

クラスターで自動フェイルオーバーが有効になっている場合、プライマリノードが変更される可能性があります。したがって、アプリケーションでノードのロールを確認し、すべての読み取りエンドポイントを更新する必要があります。これにより、プライマリに大きな負荷がかかっていないことを確認できます。自動フェイルオーバーを無効にしても、ノードのロールは変わりません。ただし、自動フェイルオーバーが有効になっているクラスターと比較して、マネージド型またはアンマネージド型のオペレーションのダウンタイムは長くなります。

読み取りリクエストの転送先を単一のリードレプリカノードに限定しないでください。そのノードが使用できなくなると、読み取りが停止する可能性があります。メンテナンス中の読み取り中断を回避するには、プライマリからの読み取れるようにフォールバックするか、少なくとも2つのリードレプリカを用意してください。

Redis のベストプラクティス

Redis を使用してパフォーマンスと信頼性を向上するためのベストプラクティスを以下に紹介します。

- クラスターモード対応の構成を使用する — クラスターモードを有効にすると、キャッシュを水平方向にスケールして、クラスターモードを無効にした構成よりもストレージとスループットを向上できます。ElastiCache サーバーレスは、クラスターモードが有効な構成でのみ使用できます。
- 存続期間の長い接続を使用する — 新しい接続の作成にはコストと時間がかかり、キャッシュの CPU リソースが消費されます。できれば (接続プーリングなどで) 接続を再利用して、こうしたコストを多くのコマンドで分担します。
- レプリカから読み取る — ElastiCache サーバーレスを使用している場合や、リードレプリカ (独自設計型クラスター) をプロビジョニングしている場合は、読み取りをレプリカに転送してスケーラビリティを向上し、レイテンシーを軽減させます。レプリカからの読み取りには、プライマリとの結果整合性があります。

独自設計型クラスターでは、読み取りリクエストの転送先を単一のリードレプリカに限定しないでください。そのノードで障害が起きると、一時的に読み取りができなくなる可能性があります。読み取りリクエストを少なくとも 2 つのリードレプリカに転送するか、1 つのレプリカとプライマリに転送するようにクライアントを設定してください。

ElastiCache サーバーレスでは、レプリカポート (6380) からの読み取りは、可能な限りクライアントのローカルアベイラビリティゾーンに転送されるため、取得レイテンシーが軽減されます。障害発生時には、自動的に他のノードにフォールバックします。

- コストの高いコマンドを避ける – KEYS や SMEMBERS コマンドのような、計算コストが高いオペレーションや入出力量の多いオペレーションを避けてください。これらのオペレーションでは、クラスターへの負荷が増えてクラスターのパフォーマンスに影響するため、これらを避けるアプローチをお勧めします。代わりに、SCAN コマンドおよび SSCAN コマンドを使用します。
- Lua のベストプラクティスに従う – 長時間実行する Lua スクリプトを避け、常に Lua スクリプトで使用されているキーを前に宣言します。Lua スクリプトがクロススロットコマンドを使用していないことを確認するために、この方法をお勧めします。Lua スクリプトで使用されるキーが同じスロットに属していることを確認します。
- シャードされた pub/sub を使用する — Redis を使用して高スループットの Pub/Sub ワークロードをサポートする場合は、[シャードされた Pub/Sub](#) (Redis 7 以降で利用可能) の使用を推奨します。クラスターモードが有効なクラスターにおける従来の Pub/Sub は、クラスター内のすべてのノードにメッセージをブロードキャストするため、EngineCPUUtilization が高くなる可能性があります。

ります。ElastiCache サーバーレスでは、従来の Pub/Sub コマンドは内部的には、シャードされた Pub/Sub コマンドを使用する点に注意してください。

キャッシュ戦略

以下のトピックでは、キャッシュを設定および維持するための戦略について説明します。

キャッシュするデータとデータへのアクセスパターンに基づいて、キャッシュを入力し維持するために実装する戦略とは何か。たとえば、ゲームサイトやトレンドのニュースのランキングトップ 10 で同じ同じ戦略は使用したくないでしょう。このセクションの後半では、一般的なキャッシュのメンテナンス戦略、利点および欠点について説明します。

トピック

- [遅延読み込み](#)
- [書き込みスルー](#)
- [TTL の追加](#)
- [関連トピック](#)

遅延読み込み

その名前が示すようため、[遅延読み込み] は、必要なときにのみキャッシュにデータを読み込むキャッシュ戦略です。これは、以下で説明するように動作します。

Amazon ElastiCacheは、インメモリ key-value ストアで、アプリケーションとアプリケーションがアクセスするデータストア (データベース) 間にあります。アプリケーションがデータをリクエストする場合は、常に ElastiCache キャッシュに最初にリクエストを行います。データがキャッシュにあり最新である場合、ElastiCache はアプリケーションにデータを返します。データがキャッシュにない場合、または期限が切れている場合は、アプリケーションはデータストアからのデータをリクエストします。その後、データストアはアプリケーションにデータを返します。次に、アプリケーションは、ストアから受信したデータをキャッシュに書き込みます。このようにして、次回リクエストされたときに、より迅速に取得できます。

[キャッシュヒット] は、データがキャッシュにあり、期限切れでない場合に発生します。

1. アプリケーションは、キャッシュに対してデータをリクエストします。
2. キャッシュはアプリケーションにデータを返します。

[キャッシュミス] は、データがキャッシュにないか、期限切れの場合に発生します。

1. アプリケーションは、キャッシュに対してデータをリクエストします。

2. キャッシュにはリクエストされたデータがないため、null を返します。
3. アプリケーションはデータベースに対してデータをリクエストし、取得します。
4. アプリケーションは、新しいデータでキャッシュを更新します。

遅延読み込みの利点と欠点

遅延読み込みの利点は次のとおりです。

- リクエストされたデータのみをキャッシュします。

ほとんどのデータがリクエストされないため、遅延読み込みではデータでキャッシュがいっぱいになることを回避できます。

- ノード障害は、アプリケーションにとって致命的ではありません。

ノードで障害が発生して新しい空のノードに置き換えられた場合、アプリケーションはレイテンシーが長くなっても機能し続けます。新規ノードへのリクエストが行われると、それぞれのキャッシュミスにより、データベースのクエリが行われます。同時に、後続のリクエストがキャッシュからデータを取得できるように、データコピーがキャッシュに追加されます。

遅延読み込みの欠点は次のとおりです。

- キャッシュミスのペナルティがあります。1回のキャッシュのミスで3回のトリップ:

1. キャッシュに対する最初のデータリクエスト
2. データベースへのデータクエリ
3. キャッシュにデータを書き込む

これらのミスにより、アプリケーションによるデータの取得に相当な遅延が発生する可能性があります。

- 古いデータ。

キャッシュミスがある場合にのみデータがキャッシュに書き込まれる場合は、キャッシュ内のデータが古くなる可能性があります。この結果は、データベースのデータが変更されたときに、キャッシュへの更新がないために発生します。この問題に対処するには、[書き込みスルー](#) および [TTL の追加](#) 戦略を使用できます。

遅延読み込み擬似コードの例

次のコードは、遅延読み込みロジックの擬似コードの例です。

```
// *****  
// function that returns a customer's record.  
// Attempts to retrieve the record from the cache.  
// If it is retrieved, the record is returned to the application.  
// If the record is not retrieved from the cache, it is  
//   retrieved from the database,  
//   added to the cache, and  
//   returned to the application  
// *****  
get_customer(customer_id)  
  
    customer_record = cache.get(customer_id)  
    if (customer_record == null)  
  
        customer_record = db.query("SELECT * FROM Customers WHERE id = {0}",  
customer_id)  
        cache.set(customer_id, customer_record)  
  
    return customer_record
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
customer_record = get_customer(12345)
```

書き込みスルー

書き込みスルー戦略では、データがデータベースに書き込まれると常にデータを追加するか、キャッシュのデータを更新します。

書き込みスルーの利点と欠点

書き込みスルーの利点は次のとおりです。

- キャッシュのデータが古くなりません。

キャッシュにデータベースにデータが書き込まれるたびにキャッシュのデータが更新されるため、キャッシュのデータが常に最新の状態になります。

- 書き込みペナルティ対読み取りペナルティ。

1 回の書き込みで 2 回のトリップ:

1. キャッシュへの書き込み
2. データベースへの書き込み

レイテンシーをプロセスに追加します。つまり、エンドユーザーは一般的に、データの取得時よりもデータの更新時のレイテンシーに対して寛容です。更新は作業量が大きく時間がかかるのが常です。

書き込みスルーの欠点は次のとおりです。

- 欠落データ。

ノード障害またはスケールアウトにより、新規ノードをスピニングすると、データが欠落しています。このデータは、データベースで追加または更新されるまで失われ続けます。これを最小限に抑えるには、[\[遅延読み込み\]](#) を書き込みスルーで指定します。

- キャッシュの変動。

ほとんどのデータは読み込まれないため、これはリソース浪費です。[\[有効期限 \(TTL\) の値を追加する\]](#) を使用すると、無駄なスペースを最小限に抑えることができます。

書き込みスルー擬似コードの例

以下は、書き込みスルーロジックの擬似コードの例です。

```
// *****  
// function that saves a customer's record.  
// *****  
save_customer(customer_id, values)  
  
    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)  
    cache.set(customer_id, customer_record)  
    return success
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
save_customer(12345, {"address": "123 Main"})
```

TTL の追加

遅延読み取りはデータが古くなる可能性があります、空ノードによる障害は発生しません。書き込みスルーでは常に新しいデータとなりますが、空ノードの障害が発生して、過剰なデータがキャッシュに入力される可能性があります。それぞれの書き込みに有効期限 (TTL) の値を追加することで、それぞれの戦略のメリットが得られます。同時に、過剰なデータでキャッシュがいっぱいになる事態が避けられます。

[有効期限 (TTL)] は、キーの有効期限までの秒数を指定する整数値です。Redis は、この値の秒またはミリ秒を指定できます。アプリケーションが期限切れのキーを読み込もうとすると、キーが見つからないものとして処理されます。データベースにキーについてクエリされ、キャッシュが更新されます。このアプローチは、値が古くなっていないことを保証するものではありません。ただし、これはデータが古くなりすぎることを防ぎ、キャッシュの値がデータベースから時々更新されることを必要とします。

詳細については、[Redis の set コマンド](#)を参照してください。

TTL 擬似コードの例

以下は、TTL のある書き込みスルーロジックの擬似コードの例です。

```
// *****  
// function that saves a customer's record.  
// The TTL value of 300 means that the record expires  
//   300 seconds (5 minutes) after the set command  
//   and future reads will have to query the database.  
// *****  
save_customer(customer_id, values)  
  
    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)  
    cache.set(customer_id, customer_record, 300)  
  
    return success
```

以下は、TTL のある遅延読み込みロジックの擬似コードの例です。

```
// *****  
// function that returns a customer's record.  
// Attempts to retrieve the record from the cache.
```

```
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//   retrieved from the database,
//   added to the cache, and
//   returned to the application.
// The TTL value of 300 means that the record expires
//   300 seconds (5 minutes) after the set command
//   and subsequent reads will have to query the database.
// *****
get_customer(customer_id)

    customer_record = cache.get(customer_id)

    if (customer_record != null)
        if (customer_record.TTL < 300)
            return customer_record          // return the record and exit function

    // do this only if the record did not exist in the cache OR
    //   the TTL was >= 300, i.e., the record in the cache had expired.
    customer_record = db.query("SELECT * FROM Customers WHERE id = {0}", customer_id)
    cache.set(customer_id, customer_record, 300) // update the cache
    return customer_record          // return the newly retrieved record and exit
function
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
save_customer(12345, {"address": "123 Main"})
```

```
customer_record = get_customer(12345)
```

関連トピック

- [インメモリデータストア](#)
- [エンジンとバージョンの選択](#)
- [Redis ElastiCache のスケーリング](#)

独自設計型クラスターの管理

このセクションでは、独自設計型クラスターの管理に役立つトピックを案内します。

Note

これらのトピックは、ElastiCache サーバーレスには適用されません。

トピック

- [Redis クラスターAuto ElastiCache Scaling](#)
- [クラスターモードの変更](#)
- [グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)
- [レプリケーショングループを使用する高可用性](#)
- [メンテナンスの管理](#)
- [パラメータグループを使用したエンジンパラメータの設定](#)

Redis クラスターAuto ElastiCache Scaling

前提条件

ElastiCache Redis 用Auto Scaling は以下のものに限定されます。

- Redis エンジンバージョン 6.0 以降を実行する Redis (クラスターモードが有効) クラスター
- Redis エンジンバージョン 7.0.7 以降を実行する データ階層化 (クラスターモードが有効) クラスター
- インスタンスサイズ - Large、XLarge、2XLarge
- インスタンスタイプファミリー - R7g、R6g、R6gd、R5、M7g、M6g、M5、C7gn
- Redis ElastiCache の Auto Scaling in は、グローバルデータストア、Outposts、またはLocal Zones で実行されているクラスターではサポートされていません。

Redis Auto Scaling ElastiCache によるキャパシティの自動管理

ElastiCache for Redis auto スケーリングは、ElastiCache for Redis サービス内の必要なシャードまたはレプリカを自動的に増減する機能です。ElastiCache for Redis は、アプリケーションの Auto Scaling サービスを活用してこの機能を提供します。詳細については、[Application Auto Scaling](#) を参照してください。自動スケーリングを使用するには、CloudWatch 割り当てたメトリクスとターゲット値を使用するスケーリングポリシーを定義して適用します。ElastiCache for Redis auto Scaling は、ポリシーを使用して実際のワークロードに応じてインスタンス数を増減します。

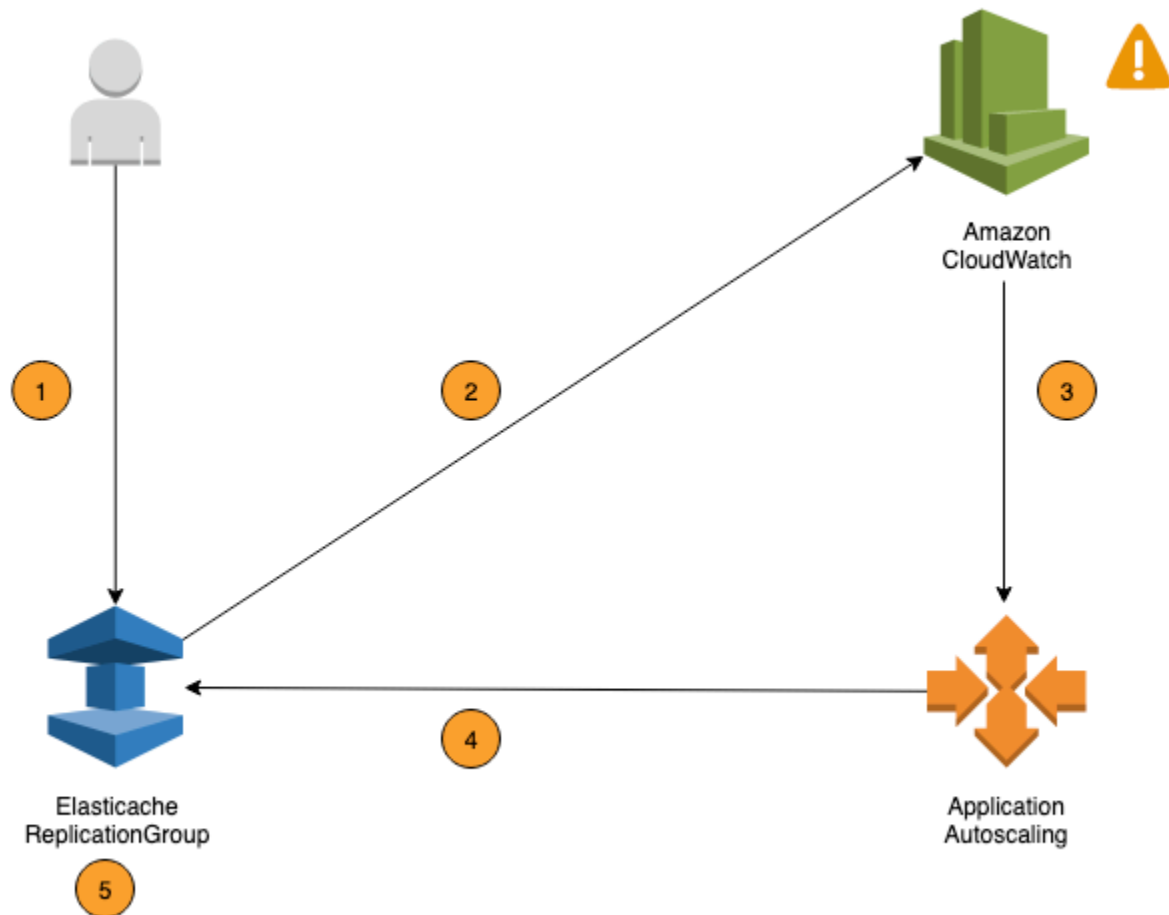
を使用して、AWS Management Console 定義済みのメトリックスに基づいてスケーリングポリシーを適用できます。predefined metric は列挙型で定義されるため、それをコード内に名前で指定するか、AWS Management Consoleで使用できます。カスタムのメトリクスは、AWS Management Consoleを使用した選択には使用できません。または、AWS CLI またはアプリケーションの Auto Scaling API を使用して、事前定義またはカスタムメトリックスに基づいてスケーリングポリシーを適用することもできます。

ElastiCache for Redis では、以下のディメンションのスケーリングをサポートしています。

- [シャード] — 手動オンラインリシャードイングと同様に、クラスター内のシャードを自動的に追加/削除します。この場合、Redis ElastiCache の auto Scaling はユーザーに代わってスケーリングをトリガーします。
- [レプリカ] – 手動によるレプリカの増加/減少オペレーションと同様に、クラスター内のレプリカを自動的に追加/削除します。ElastiCache Redis の auto Scaling では、クラスター内のすべてのシャードでレプリカを均一に追加/削除します。

ElastiCache for Redis は以下のタイプの自動スケーリングポリシーをサポートしています。

- [ターゲット追跡スケーリングポリシー](#) – 特定のメトリクスのターゲット値に基づいて、サービスが実行するシャード/レプリカの数を増減させます。これはサーモスタットが家の温度を維持する方法に似ています。温度を選択すれば、後はサーモスタットがすべてを実行します。
- [Application ElastiCache for Redis auto Scaling のスケジュールスケーリング](#) — 日付と時刻に基づいて、サービスが実行するシャード/レプリカの数を増減します。



次の手順は、前の図に示した Redis ElastiCache 用の auto Scaling プロセスをまとめたものです。

1. ElastiCache for Redis レプリケーショングループ用の for Redis auto スケーリングポリシーを作成します。ElastiCache
2. ElastiCache Redis の場合、auto スケーリングはユーザーに代わって 2 CloudWatch つのアラームを作成します。各ペアはメトリクスの上限と下限を示します。CloudWatch これらのアラームは、クラスターの実際の使用率が一定期間目標の使用率から逸脱したときにトリガーされます。コンソールでアラームを表示できます。
3. 設定したメトリクス値が特定の時間にわたって目標使用率を超えた (または目標を下回った) 場合、Redis auto Scaling CloudWatch ElastiCache を呼び出してスケーリングポリシーを評価するアラームがトリガーされます。
4. ElastiCache Redis の auto Scaling では、クラスターの容量を調整するための変更リクエストが発行されます。

5. ElastiCache for Redis は変更リクエストを処理し、クラスターのシャード/レプリカの容量を目標の使用率に近づけるように動的に増加 (または減少) します。

Redis Auto Scaling ElastiCache の仕組みを理解するために、UsersClusterという名前のクラスターがあると仮定します。CloudWatch のメトリクスをモニタリングすることでUsersCluster、トラフィックがピークのときにクラスターが必要とする最大シャード数と、トラフィックが最も低いときの最小シャード数を決定します。また、UsersCluster クラスターの CPU 使用率のターゲット値を決定します。ElastiCache for Redis auto Scaling は、ターゲットトラッキングアルゴリズムを使用して、使用率が目標値またはそれに近い値に維持されるように、UsersClusterプロビジョニングされたのシャードを必要に応じて調整します。

Note

スケーリングにはかなりの時間がかかり、シャードのバランスを取り戻すには追加のクラスターリソースが必要になります。ElastiCache for Redis Auto Scaling は、実際のワークロードが数分間持続的に上昇 (または低下) している場合にのみリソース設定を変更します。ElastiCache for Redis の auto Scaling ターゲットトラッキングアルゴリズムは、長期にわたってターゲットの使用率を選択した値またはそれに近い値に保つことを目指します。

Auto Scaling ポリシー

スケーリングポリシーには、次のコンポーネントがあります。

- ターゲットメトリクス – ElastiCache for Redis の Auto Scaling がスケーリングするタイミングと量を判断するために使用する CloudWatch のメトリクス。
- 最小容量と最大容量 – スケーリングに使用されるシャードまたはレプリカの最小数および最大数。

Important

Auto Scaling ポリシーの作成中に、現在の容量が設定された最大容量よりも大きい場合、ポリシーの作成時に maxCapacity にスケールインします。同様に、現在の容量が設定された最少容量よりも小さい場合は、最小容量にスケールアウトします。

- クールダウン期間 – スケールインまたはスケールアウトアクティビティが完了してから別のスケールアウトアクティビティが開始されるまでの時間 (秒)。

- サービスにリンクされたロール — AWS Identity and Access Management (IAM) ロールは、特定の AWS サービスにリンクされています。サービスにリンクされたロールは、ユーザーに代わってサービスから AWS の他のサービスを呼び出すために必要なすべてのアクセス許可を備えています。ElastiCache for Redis の Auto Scaling は、ユーザーに代わって自動的にこの `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG` というロールを生成します。
- スケールインアクティビティの有効化または無効化 - ポリシーに対してスケールインアクティビティを有効化または無効化できます。

トピック

- [Auto Scaling のターゲットメトリックス](#)
- [最小容量と最大容量](#)
- [クールダウン期間](#)
- [スケールインアクティビティの有効化または無効化](#)

Auto Scaling のターゲットメトリックス

このタイプのポリシーでは、ターゲット追跡スケールリングポリシー設定で、事前定義されたメトリックスまたはカスタムメトリックスとそのメトリックスのターゲット値を指定します。ElastiCache for Redis の Auto Scaling は、スケールリングポリシーをトリガーする CloudWatch アラームを作成および管理し、メトリックスとターゲット値に基づいてスケールリング調整値を計算します。スケールリングポリシーは、指定されたターゲット値、またはそれに近い値にメトリックスを維持するため、必要に応じてシャード/レプリカを追加または削除します。メトリックスをターゲット値に近い値に維持することに加えて、ターゲット追跡スケールリングポリシーは、変化するワークロードによるメトリックスの変動に適応します。そのようなポリシーは、クラスターに使用可能なシャード/レプリカ数の急速な変動の最小化もします。

たとえば、事前定義された平均 `ElastiCachePrimaryEngineCPUUtilization` メトリックスを使用するスケールリングポリシーを考慮してください。そのようなポリシーは、70 パーセントなどの指定された使用率に、またはそれに近い割合に CPU 使用率を維持できます。

Note

各クラスターについては、各ターゲットメトリックスに対して 1 つの Auto Scaling ポリシーのみを作成できます。

最小容量と最大容量

シャード

ElastiCache for Redis の Auto Scaling で ElastiCache がスケーリングできるシャードの最大数を指定できます。この値は、250 以下で、最小 1 である必要があります。ElastiCache for Redis の Auto Scaling で ElastiCache が管理するシャードの最小数も指定できます。この値は 1 以上で、最大シャードで指定された値である 250 以下である必要があります。

レプリカ

ElastiCache for Redis の Auto Scaling が管理するレプリカの最大数を指定できます。この値は、1 以下にする必要があります。ElastiCache for Redis の Auto Scaling が管理するレプリカの最小数も指定できます。この値は 1 以上で、最大レプリカで指定された値である 5 以下である必要があります。

通常のトラフィックに必要なシャード/レプリカの最小数と最大数を決定するには、モデルに対するトラフィックの予想レートで Auto Scaling の設定をテストします。

Note

ElastiCache for Redis の Auto Scaling ポリシーは、定義した最大サイズに達するまで、またはサービス制限が適用されるまで、クラスター容量を増やします。この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

Important

トラフィックがないときにスケールインするバリエーションのトラフィックがゼロになった場合、ElastiCache for Redis は、指定されたインスタンスの最小数に自動的にスケールインします。

クールダウン期間

クラスターのスケールインやスケールアウトに影響するクールダウン期間を追加することで、ターゲット追跡スケーリングポリシーの応答性を調整できます。クールダウン期間を設定すると、その期間が過ぎるまでその後のスケールインやスケールアウトのリクエストがブロックされます。これにより、スケールインリクエストのための ElastiCache for Redis クラスターでのシャード/レプリカの削

除、およびスケールアウトリクエストのためのシャード/レプリカの作成を遅らせます。以下のクールダウン期間を指定できます。

- スケールインアクティビティは、ElastiCache for Redis クラスターのシャード/レプリカ数を減らします。スケールインのクールダウン期間は、スケールインアクティビティが完了してから別のスケールインアクティビティが開始されるまでの時間 (秒) を指定します。
- スケールアウトアクティビティは、ElastiCache for Redis クラスターのシャード/レプリカ数を増やします。スケールアウトのクールダウン期間は、スケールアウトアクティビティが完了してから別のスケールアウトアクティビティが開始されるまでの時間 (秒) を指定します。

スケールインやスケールアウトのクールダウン期間が指定されない場合、スケールアウトのデフォルトは 600 秒で、スケールインのデフォルトは 900 秒です。

スケールインアクティビティの有効化または無効化

ポリシーに対してスケールインアクティビティを有効化または無効化できます。スケールインアクティビティを有効にすると、スケーリングポリシーはシャード/レプリカを削除できます。スケールインアクティビティが有効な場合、スケーリングポリシーのスケールインのクールダウン期間がスケールインアクティビティに適用されます。スケールインアクティビティを無効にすると、スケーリングポリシーはシャード/レプリカを削除できなくなります。

Note

スケールアウトアクティビティは、スケーリングポリシーが必要に応じて ElastiCache for Redis のシャード/レプリカを作成できるように、常に有効にしておきます。

Redis Auto Scaling ElastiCache に必要な IAM 権限

ElastiCache for Redis Auto Scaling は、ElastiCache for Redis と CloudWatch、Application Auto Scaling API の組み合わせによって可能になります。クラスターは ElastiCache for Redis で作成および更新され、アラームは作成され CloudWatch、スケーリングポリシーは Application Auto Scaling で作成されます。クラスターを作成および更新するための標準の IAM 権限に加えて、Redis Auto Scaling 設定にアクセスする ElastiCache IAM ユーザーには、動的スケーリングをサポートするサービスに対する適切な権限が必要です。IAM ユーザーには、次のポリシー例に示すアクションを使用するためのアクセス許可が必要です。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "application-autoscaling:*",
      "elasticache:DescribeReplicationGroups",
      "elasticache:ModifyReplicationGroupShardConfiguration",
      "elasticache:IncreaseReplicaCount",
      "elasticache:DecreaseReplicaCount",
      "elasticache:DescribeCacheClusters",
      "elasticache:DescribeCacheParameters",
      "cloudwatch:DeleteAlarms",
      "cloudwatch:DescribeAlarmHistory",
      "cloudwatch:DescribeAlarms",
      "cloudwatch:DescribeAlarmsForMetric",
      "cloudwatch:GetMetricStatistics",
      "cloudwatch:ListMetrics",
      "cloudwatch:PutMetricAlarm",
      "cloudwatch:DisableAlarmActions",
      "cloudwatch:EnableAlarmActions",
      "iam:CreateServiceLinkedRole",
      "sns:CreateTopic",
      "sns:Subscribe",
      "sns:Get*",
      "sns:List*"
    ],
    "Resource": "arn:aws:iam::123456789012:role/autoscaling-roles-for-cluster"
  }
]
```

サービスリンクロール

ElastiCache for Redis auto Scaling サービスには、CloudWatch クラスターとアラームを記述する権限と、ユーザーに代わって ElastiCache for Redis のターゲット容量を変更する権限も必要です。ElastiCache for Redis クラスターで Auto Scaling を有効にすると、という名前のサービスにリンクされたロールが作成されます。AWSServiceRoleForApplicationAutoScaling_ElastiCacheRGこのサービスにリンクされたロールは、ポリシーのアラームを記述したり、フリートの現在の容量を監視したり、フリートの容量を変更したりするための権限を Redis auto Scaling ElastiCache に付与します。サービスにリンクされたロールは、Redis auto Scaling ElastiCache のデフォルトロールです。詳細については、

『Application Auto Scaling Auto Scaling ユーザーガイド』の「[Redis auto Scaling ElastiCache 用のサービスにリンクされたロール](#)」を参照してください。

Auto Scaling のベストプラクティス

Auto Scaling に登録する前に、以下のことをお勧めします。

1. 追跡メトリクスを 1 つだけ使用 — クラスターに CPU 負荷の高いワークロードまたはデータ集約型のワークロードがあるかどうかを識別し、対応する定義済みメトリックを使用してスケーリングポリシーを定義します。
 - エンジン CPU: `ElastiCachePrimaryEngineCPUUtilization` (シャードディメンション) または `ElastiCacheReplicaEngineCPUUtilization` (レプリカディメンション)
 - データベースの使用状況:
`ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` このスケーリングポリシーは、クラスターで `maxmemory-policy` が `noeviction` に設定されている場合に最適です。

クラスターのディメンションごとに複数のポリシーを使用することは避けることをお勧めします。ElastiCache Redis の場合、Auto Scaling は、スケールアウトの準備ができていないターゲット追跡ポリシーがある場合はスケラブルターゲットをスケールアウトしますが、すべてのターゲット追跡ポリシー (スケールイン部分が有効になっている) がスケールインできる状態になった場合にのみスケールインします。複数のポリシーによって、スケラブルなターゲットが同時にスケールアウトまたはスケールインするように指示される場合、Auto Scaling は、スケールインとスケールアウトの両方で最大の容量を提供するポリシーに基づいてスケールします。

2. ターゲット追跡のカスタマイズされたメトリクス — Target Tracking 用にカスタマイズされたメトリクスを使用する場合は注意が必要です。Auto Scaling は、ポリシー用に選択されたメトリクスの変更に比例してスケールアウトするのに最適です。スケーリングアクションに比例して変更されないメトリクスがポリシーの作成に使用されると、可用性やコストに影響する可能性のあるスケールアウトまたはスケールインアクションが継続する可能性があります。

データ階層化クラスター (r6gd ファミリーのインスタンスタイプ) では、スケーリングにメモリのメトリクスを使用しないでください。

3. スケジュールに基づくスケーリング — ワークロードが確定的 (特定の時点で高/低に達する) であることが判明した場合は、スケジュールされたスケーリングを使用し、必要に応じてターゲット容量を設定することをお勧めします。ターゲット追跡は、非決定的なワークロードや、必要なターゲットメトリクスでクラスターを操作する場合に最適です。これにより、より多くのリソースが必要な場合はスケールアウトし、必要な場合はスケールインします。

4. スケールインを無効化する — ターゲット追跡での Auto Scaling は、ワークロードが徐々に増減するクラスターに最適です。メトリクスのスパイク/ディップが連続するスケールアウト/イン振動を引き起こす可能性があるためです。このような振動を避けるために、スケールインを無効にして開始し、後でいつでも必要に応じて手動でスケールインすることができます。
5. アプリケーションをテスト — 可用性の問題を回避するために、スケーリングポリシーを作成しながら、クラスターに必要な最小/最大シャード/レプリカの絶対値を決定するために、最小/最大ワークロードを推定してアプリケーションをテストすることをお勧めします。Auto Scaling は Max にスケールアウトし、ターゲットに設定された最小しきい値にスケールインできます。
6. 目標値の定義 — 4 CloudWatch 週間にわたるクラスター使用率に対応するメトリクスを分析して、目標値のしきい値を決定できます。選択する値が不明な場合は、サポートされる最小定義メトリクス値から開始することをお勧めします。
7. AutoScaling On Target Tracking は、シャード/レプリカのディメンションにわたってワークロードが均一に分散されているクラスターに最適です。不均一な分布を持つと、次のことが可能になります。
 - いくつかのホットシャード/レプリカでワークロードの急増/減少が原因で、必要のない場合のスケーリング。
 - ホットシャード/レプリカがあるにもかかわらず、全体的な平均ターゲットに近いために必要なときにスケーリングされません。

Note

ElastiCache クラスターをスケールアウトすると、既存のノード (ランダムに選択) の 1 つにロードされた関数が新しいノードに自動的に複製されます。クラスターに Redis 7.0 以上があり、アプリケーションで [Redis 関数](#) を使用している場合は、スケールアウトする前に、クラスターがシャードにより異なる関数定義にならないように、すべての関数をすべてのシャードに読み込むことをお勧めします。

に登録したら AutoScaling、次の点に注意してください。

- Auto Scaling でサポートされる設定には制限があるため、Auto Scaling に登録されているレプリケーショングループの設定を変更しないことをお勧めします。次に例を示します。
 - インスタンスタイプをサポートされていないタイプに手動で変更します。
 - レプリケーショングループをグローバルデータストアに関連付けます。
 - ReservedMemoryPercent パラメータの変更。

- ポリシーの作成時に設定された Min/Max 容量を超えるシャード/レプリカを手動で増減します。

シャードでの Auto Scaling の使用

以下では、ターゲット追跡とスケジュールされたポリシーの詳細と、AWS Management Console、AWS CLI および API を使用してそれらを適用する方法を説明します。

ターゲット追跡スケーリングポリシー

ターゲット追跡スケーリングポリシーで、メトリクスを選択してターゲット値を設定します。ElastiCache for Redis の Auto Scaling は、スケーリングポリシーをトリガーする CloudWatch アラームを作成および管理し、メトリクスとターゲット値に基づいてスケーリング調整値を計算します。スケーリングポリシーは、指定されたターゲット値、またはそれに近い値にメトリクスを維持するため、必要に応じてシャードを追加または削除します。ターゲットの追跡スケーリングポリシーは、メトリクスをターゲット値近くに維持することに加えて、負荷パターンの変動によるメトリクスの変動に合わせて調整し、フリートの容量の急速な変動を最小化します。

たとえば、設定されたターゲット値を持つ事前定義された平均

ElastiCachePrimaryEngineCPUUtilization メトリクスを使用するスケーリングポリシーを考慮してください。このようなポリシーは、指定されたターゲット値、またはそれに近い値に CPU 使用率を維持できます。

事前定義メトリクス

定義済みメトリクスとは、特定の CloudWatch メトリクスの特定の名前、ディメンション、統計 (average) を参照する構造です。Auto Scaling ポリシーでは、クラスターの次の事前定義されたメトリクスのいずれかを定義します。

事前定義済みメトリクス名	CloudWatch メトリクス名	CloudWatch メトリクスディメンション	不適格なインスタンスタイプ
ElastiCachePrimaryEngineCPUUtilization	EngineCPUUtilization	ReplicationGroupId、ロール = プライマリ	なし

事前定義済みメトリクス名	CloudWatch メトリクス名	CloudWatch メトリクスディメンション	不適格なインスタンスタイプ
ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage	DatabaseCapacityUsageCountedForEvictPercentage	Redis レプリケーショングループメトリクス	なし
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage	Redis レプリケーショングループメトリクス	R6gd

これらのインスタンスタイプはメモリと SSD の両方にデータを保存するため、データ階層型インスタンスタイプでは `ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage` を使用できません。データ階層インスタンスの想定される使用例は、メモリを 100% 使用し、必要に応じて SSD をいっぱいにすることです。

シャードの Auto Scaling 基準

事前定義されたメトリクスが Target 設定以上であることを検出した場合、シャードの容量が自動的に増加します。ElastiCache for Redis は、ターゲットからの変動パーセントと現在のシャードの 20 パーセントの 2 つの数字のうち、大きい方の数字に等しいカウントでクラスターシャードをスケールアウトします。スケールインの場合、全体的なメトリクス値が定義されたターゲットの 75% を下回らない限り、ElastiCache for Redis は自動スケールインしません。

スケールアウトの例では、50個のシャードを持っている場合

- ターゲットが 30% 超えた場合、ElastiCache for Redis は 30% スケールアウトして、クラスターあたり 65 個のシャードになります。

- ターゲットが 10% 超えた場合、ElastiCache for Redis はデフォルトで最小値 20% でスケールアウトして、クラスターあたり 60 個のシャードになります。

スケールインの例で、ターゲット値として 60% を選択した場合、ElastiCache for Redis は、メトリクスが 45% 以下 (ターゲットの 60% を 25% 下回る) になるまで自動スケールインしません。

Auto Scaling に関する考慮事項

次の考慮事項に注意が必要です。

- ターゲットの追跡スケールリングポリシーでは、指定されたメトリクスがターゲット値を超えている場合、スケールアウトする必要があると見なされます。指定されたメトリクスがターゲット値を下回っている場合、ターゲットの追跡スケールリングポリシーを使用してスケールアウトすることはできません。ElastiCache for Redis は、クラスター内の既存のシャードのターゲットの最小偏差の 20% 分だけシャードをスケールアウトします。
- 指定されたメトリクスに十分なデータがない場合、ターゲットの追跡スケールリングポリシーによってスケールされません。不十分なデータの利用率は低いと解釈されないため、スケールインされません。
- ターゲット値と実際のメトリクスデータポイント間にギャップが発生する場合があります。これは、ElastiCache for Redis の Auto Scaling が追加または削除する容量を決定するときに、その数を切り上げまたは切り捨てて常に控えめに動作するためです。これにより、不十分な容量を追加したり、必要以上に容量を削除することを防ぎます。
- アプリケーションの可用性を高めるために、サービスのスケールアウトはメトリクスに比例して可能な限り高速に行われますが、スケールインはより抑制されています。
- それぞれが異なるメトリクスを使用していれば、ElastiCache for Redis クラスターに対して複数のターゲット追跡スケールリングポリシーを設定できます。ElastiCache for Redis の Auto Scaling の目的は常に可用性を優先することであるため、その動作は、ターゲット追跡ポリシーでスケールアウトまたはスケールインの準備ができていないかによって異なります。ターゲット追跡ポリシーのいずれかでスケールアウトする準備ができると、サービスがスケールアウトされますが、すべてのターゲット追跡ポリシー (スケールイン部分が有効) でスケールインする準備ができていない場合のみスケールインされます。
- ターゲットの追跡スケールリングポリシーのために ElastiCache for Redis の Auto Scaling が管理する CloudWatch アラームを編集または削除しないでください。ElastiCache for Redis の Auto Scaling では、スケールリングポリシーを削除するときに、アラームが自動的に削除されます。
- ElastiCache for Redis の Auto Scaling により、クラスターシャードを手動で変更できなくなることはありません。これらの手動調整は、スケールリングポリシーにアタッチされている CloudWatch

アラームに影響しませんが、これらの CloudWatch アラームをトリガーする可能性のあるメトリクスに影響する可能性があります。

- Auto Scaling によって管理されるこれらの CloudWatch アラームは、クラスター内のすべてのシャードでの AVG メトリクスで定義されます。したがって、ホットシャードを持つと、次のいずれかのシナリオが発生する可能性があります。
 - CloudWatch アラームをトリガーするいくつかのホットシャードへの負荷が原因で、必要のない場合にスケーリングする
 - アラームが違反しないように影響を及ぼすすべてのシャードで集約された AVG が原因で、必要な場合にスケーリングしない。
- クラスターごとのノードに対する ElastiCache for Redis のデフォルト制限は引き続き適用されます。したがって、Auto Scaling を選択するとき、最大ノード数がデフォルトの制限を超えると予測される場合は、[\[AWS サービス制限\]](#) で制限の増加をリクエストし、制限タイプ [インスタンスタイプごとのクラスターあたりのノード] を選択します。
- スケールアウト時に必要な、VPC で十分な ENI (Elastic Network Interfaces) が使用可能であることを確認します。詳細については、「[Elastic Network Interface](#)」を参照してください。
- EC2 の容量が十分でない場合、ElastiCache for Redis Auto Scaling は、容量が利用可能になるまで、スケールアウトせず、遅延します。
- ElastiCache for Redis の Auto Scaling は、シリアル化後のアイテムサイズが 256 MB を超えるスロットを持つシャードを削除しません。
- スケールイン中に、結果として得られるシャード設定で利用可能なメモリが不足している場合、シャードは削除されません。

スケーリングポリシーの追加

AWS Management Console を使用してスケーリングポリシーを追加できます。

Auto Scaling ポリシーを ElastiCache for Redis クラスターに追加するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します (クラスター名の左にあるボタンではなく、クラスター名を選択)。
4. [Auto Scaling ポリシー] タブを選択します。
5. [add dynamic scaling] (動的なスケーリングを追加) を選択します。

6. [Policy Name] で、ポリシーの名前を入力します。
7. [スケーラブルなディメンション] で、[シャード] を選択します。
8. ターゲットメトリクスには、以下のいずれかを選択します。
 - 平均 CPU 使用率に基づいてポリシーを作成するための [プライマリ CPU 使用率]。
 - データベース平均メモリに基づいてポリシーを作成するための [メモリ]。
 - データベース平均メモリに基づいてポリシーを作成するための [容量]。容量メトリクスには、データ階層化インスタンスのメモリと SSD の使用率、および他のすべてのインスタンスタイプのメモリ使用率が含まれます。
9. 目標値は、35 以上、70 以下の値を選択します。Auto Scaling は、選択したターゲットメトリクスについて、ElastiCache シャード全体でこの値を維持します。
 - プライマリCPU使用率: プライマリノードの EngineCPUUtilization メトリクスの目標値を維持します。
 - メモリ: DatabaseMemoryUsageCountedForEvictPercentage メトリクスの目標値を維持します。
 - 容量は DatabaseCapacityUsageCountedForEvictPercentage メトリクスの目標値を維持し、

クラスターシャードが追加または削除され、メトリクスが指定された値に近い値に維持されます。
10. (オプション) スケールインまたはスケールアウトのクールダウン期間は、コンソールからはサポートされていません。AWS CLI を使用して、クールダウン値を変更します。
11. [Minimum capacity (最小容量)] では、ElastiCache for Redis の Auto Scaling ポリシーが維持する必要があるシャードの最小数を入力します。
12. [Maximum capacity (最大容量)] では、ElastiCache for Redis の Auto Scaling ポリシーが維持する必要があるシャードの最大数を入力します。この値は、250 以下にする必要があります。
13. [Create] (作成) を選択します。

スケーラブルなターゲットの登録

Auto Scaling を ElastiCache for Redis クラスターで使用可能にする前に、クラスターを ElastiCache for Redis の Auto Scaling に登録します。これは、そのクラスターに適用するスケーリングのディメンションと制限を定義するためです。ElastiCache for Redis の Auto Scaling は、ElastiCache

for Redis クラスターを、クラスターシャードの数を表す `elasticache:replication-group:NodeGroups` スケーラブルディメンションで動的にスケールリングします。

AWS CLI の使用

ElastiCache for Redis クラスターを登録するには、[register-scalable-target](#) コマンドを次のパラメータをとともに使用します。

- `--service-namespace` – この値は `elasticache` に設定します。
- `--resource-id` — ElastiCache for Redis クラスターのリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、例えば `replication-group/myscalablecluster` です。
- `--scalable-dimension` – この値は `elasticache:replication-group:NodeGroups` に設定します。
- `--max-capacity` — ElastiCache for Redis の Auto Scaling で管理するシャードの最大数。 `--min-capacity`、`--max-capacity`、およびクラスター内のシャードの数の関係については、「[最小容量と最大容量](#)」を参照してください。
- `--min-capacity` — ElastiCache for Redis の Auto Scaling で管理するシャードの最小数。 `--min-capacity`、`--max-capacity`、およびクラスター内のシャードの数の関係については、「[最小容量と最大容量](#)」を参照してください。

Example

以下の例では、`myscalablecluster` という名前の ElastiCache for Redis クラスターを登録します。この登録は、クラスターが 1 から 10 個のシャードを持つよう動的にスケールされることを示します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling register-scalable-target \  
  --service-namespace elasticache \  
  --resource-id replication-group/myscalablecluster \  
  --scalable-dimension elasticache:replication-group:NodeGroups \  
  --min-capacity 1 \  
  --max-capacity 10 \  

```

Windows の場合:

```
aws application-autoscaling register-scalable-target ^
  --service-namespace elasticache ^
  --resource-id replication-group/myscalablecluster ^
  --scalable-dimension elasticache:replication-group:NodeGroups ^
  --min-capacity 1 ^
  --max-capacity 10 ^
```

API の使用

ElastiCache クラスターを登録するには、[register-scalable-target](#) コマンドを次のパラメータとともに使用します。

- ServiceNamespace – この値は `elasticache` に設定します。
- ResourceID — ElastiCache クラスターのリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、たとえば `replication-group/myscalablecluster` です。
- ScalableDimension — この値は `elasticache:replication-group:NodeGroups` に設定します。
- MinCapacity — ElastiCache for Redis の Auto Scaling で管理するシャードの最小数。—`min-capacity`、—`max-capacity`、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。
- MaxCapacity — ElastiCache for Redis の Auto Scaling で管理するシャードの最大数。—`min-capacity`、—`max-capacity`、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。

Example

以下の例では、`myscalablecluster` という名前の ElastiCache for Redis クラスターをアプリケーションの Auto Scaling API に登録します。この登録は、クラスターが 1~5 個のレプリカを持つよう動的にスケールされることを示します。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
```

```
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:NodeGroups",
  "MinCapacity": 1,
  "MaxCapacity": 5
}
```

スケーリングポリシーの定義

ターゲット追跡スケーリングポリシー設定は、メトリクスとターゲット値が定義されている JSON ブロックで表されます。JSON ブロックとしてスケーリングポリシー設定をテキストファイルに保存できます。そのテキストファイルは、AWS CLI または Application Auto Scaling API を呼び出すときに使用します。ポリシー設定構文の詳細については、『Application Auto Scaling API リファレンス』の[TargetTrackingScalingPolicyConfiguration](#)を参照してください。

ターゲット追跡スケーリングポリシー設定を定義するには、次のオプションを使用できます。

トピック

- [事前定義メトリクスの使用](#)
- [カスタムメトリクスの使用](#)
- [クールダウン期間の使用](#)
- [スケールインアクティビティの無効化](#)
- [スケーリングポリシーの適用](#)

事前定義メトリクスの使用

定義済みのメトリクスを使用することで、for Redis Auto Scaling のターゲットトラッキングと連携する ElastiCache for Redis クラスターのターゲット追跡スケーリングポリシーをすばやく定義できます。ElastiCache

現在、ElastiCache Redis では Redis NodeGroup Auto Scaling ElastiCache で以下の定義済みメトリクスをサポートしています。

- `ElastiCachePrimaryEngineCPUUtilization` — for Redis EngineCPUUtilization CloudWatch クラスター内のすべてのプライマリノードにおけるメトリクスの平均値。ElastiCache

- `ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage`— ElastiCache for Redis `DatabaseMemoryUsageCountedForEvictPercentage CloudWatch` クラスター内のすべてのプライマリノードにおけるメトリックの平均値。
- `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage`— ElastiCache for Redis `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage CloudWatch` クラスター内のすべてのプライマリノードにおけるメトリックの平均値。

`EngineCPUUtilization` と `DatabaseMemoryUsageCountedForEvictPercentage`、および `DatabaseCapacityUsageCountedForEvictPercentage` メトリックスの詳細については、「[CloudWatch メトリックスの使用状況のモニタリング](#)」を参照してください。スケーリングポリシーで事前定義メトリックスを使用するには、スケーリングポリシーのターゲット追跡構成を作成します。この設定には、`PredefinedMetricSpecification TargetValue` 事前定義済みのメトリック用とそのメトリックスのターゲット値用の `a` を含める必要があります。

Example

次の例は、for Redis クラスターのターゲット追跡スケーリングの一般的なポリシー設定を示しています。ElastiCache この設定では、`ElastiCachePrimaryEngineCPUUtilization` 事前定義されたメトリックスを使用して、クラスター内のすべてのプライマリノードの CPU 平均使用率が 40% になるように ElastiCache for Redis クラスターを調整します。

```
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
  {
    "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
  }
}
```

カスタムメトリックスの使用

カスタムメトリックスを使用することで、カスタム要件を満たすターゲット追跡スケーリングポリシーを定義できます。ElastiCache スケーリングに比例して変化するあらゆるメトリックスに基づいてカスタムメトリックスを定義できます。ElastiCache すべてのメトリックスがターゲットトラッキングに役立つわけではありません。メトリックスは、有効な使用率メトリックスで、インスタンスの使用頻度を示す必要があります。クラスター内のシャードの数に比例してメトリックスの値を増減する必要があります。この比例的な増加または減少は、比例的にスケールアウトするため、またはシャードの数にメトリックスデータを使用するために必要です。

Example

次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この構成では、カスタムメトリックスは、という名前のクラスター内のすべてのシャードの平均 CPU 使用率が 50% になるように調整します。ElastiCache my-db-cluster

```
{
  "TargetValue": 50,
  "CustomizedMetricSpecification":
  {
    "MetricName": "EngineCPUUtilization",
    "Namespace": "AWS/ElastiCache",
    "Dimensions": [
      {
        "Name": "RelicationGroup","Value": "my-db-cluster"
      },
      {
        "Name": "Role","Value": "PRIMARY"
      }
    ],
    "Statistic": "Average",
    "Unit": "Percent"
  }
}
```

クールダウン期間の使用

ScaleOutCooldown の値を秒単位で指定して、クラスターをスケールアウトするためのクールダウン期間を追加することができます。同様に、ScaleInCooldown の値を秒単位で追加して、クラスターをスケールインするためのクールダウン期間を追加することができます。詳細については、「Application Auto Scaling API リファレンス」の[TargetTrackingScalingPolicyConfiguration](#)を参照してください。

次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この設定では、ElastiCachePrimaryEngineCPUUtilization事前定義されたメトリックスを使用して ElastiCache for Redis クラスターを、そのクラスター内のすべてのプライマリノードの平均 CPU 使用率 40% に基づいて調整します。この設定では、10 分間のスケールインのクールダウン期間と 5 分間のスケールアウトのクールダウン期間が提供されます。

```
{
  "TargetValue": 40.0,
```



```
"PredefinedMetricSpecification":
{
  "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
},
"ScaleInCooldown": 600,
"ScaleOutCooldown": 300
}
```

スケールインアクティビティの無効化

スケールインアクティビティを無効にすることで、ターゲット追跡スケーリングポリシー設定が ElastiCache for Redis クラスターにスケーリングされないようにすることができます。スケールインアクティビティを無効にすると、スケーリングポリシーによってシャードが削除されることなく、スケーリングポリシーによって必要に応じて作成されます。

DisableScaleIn のブール値を指定して、クラスターのアクティビティのスケールを有効または無効にすることができます。詳細については、「Application Auto Scaling API リファレンス」の [TargetTrackingScalingPolicyConfiguration](#) を参照してください。

次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この設定では、ElastiCachePrimaryEngineCPUUtilization 事前定義されたメトリックが ElastiCache、クラスター内のすべてのプライマリノードの CPU 平均使用率 40% に基づいて Redis クラスターを調整します。この設定では、スケーリングポリシーのスケールインアクティビティが無効になります。

```
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
  {
    "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
  },
  "DisableScaleIn": true
}
```

スケーリングポリシーの適用

クラスターを Redis auto Scaling に登録し、スケーリングポリシーを定義したら、登録したクラスターにスケーリングポリシーを適用します。ElastiCache Redis ElastiCache 用クラスターにスケーリングポリシーを適用するには、AWS CLI またはアプリケーションの Auto Scaling API を使用できます。

を使用してスケーリングポリシーを適用します。AWS CLI

ElastiCache for Redis クラスターにスケーリングポリシーを適用するには、[put-scaling-policy](#)以下のパラメーターを指定してコマンドを実行します。

- `--policy-name` – スケーリングポリシーの名前。
- `--policy-type` – この値は `TargetTrackingScaling` に設定します。
- `--resource-id` — Redis 用のリソース識別子。ElastiCache このパラメーターでは、リソースタイプは、たとえば `for Redis ReplicationGroup` クラスターの名前で、一意の識別子は `ElastiCache for Redis` クラスターの名前です。 `replication-group/myscalablecluster`
- `--service-namespace` – この値は `elasticache` に設定します。
- `--scalable-dimension` – この値は `elasticache:replication-group:NodeGroups` に設定します。
- `--target-tracking-scaling-policy-configuration` — `for Redis` クラスターに使用するターゲットトラッキングスケーリングポリシー設定です。ElastiCache

次の例では、`myscalablepolicy`という名前のターゲット追跡スケーリングポリシーを `for Redis Auauto Scaling ElastiCache` という名前の `for Redis` クラスターに適用します。`myscalablecluster ElastiCache` そのためには、`config.json` という名前のファイルに保存されているポリシー設定を使用します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling put-scaling-policy \  
  --policy-name myscalablepolicy \  
  --policy-type TargetTrackingScaling \  
  --resource-id replication-group/myscalablecluster \  
  --service-namespace elasticache \  
  --scalable-dimension elasticache:replication-group:NodeGroups \  
  --target-tracking-scaling-policy-configuration file://config.json
```

Windows の場合:

```
aws application-autoscaling put-scaling-policy ^  
  --policy-name myscalablepolicy ^  
  --policy-type TargetTrackingScaling ^  
  --resource-id replication-group/myscalablecluster ^
```

```
--service-namespace elasticache ^
--scalable-dimension elasticache:replication-group:NodeGroups ^
--target-tracking-scaling-policy-configuration file://config.json
```

API を使用したスケーリングポリシーの適用

ElastiCache for Redis クラスターにスケーリングポリシーを適用するには、[PutScalingPolicy](#) AWS CLI 以下のパラメーターを指定してコマンドを実行します。

- `--policy-name` – スケーリングポリシーの名前。
- `--resource-id` — Redis 用のリソース識別子。ElastiCache このパラメーターでは、リソースタイプは、たとえば `for Redis ReplicationGroup` クラスターの名前で、一意の識別子は `ElastiCache for Redis` クラスターの名前です。 `replication-group/myscalablecluster`
- `--service-namespace` – この値は `elasticache` に設定します。
- `--scalable-dimension` – この値は `elasticache:replication-group:NodeGroups` に設定します。
- `--target-tracking-scaling-policy-configuration` — `for Redis` クラスターに使用するターゲットトラッキングスケーリングポリシー設定です。ElastiCache

次の例では、`myscalablepolicy`という名前のターゲット追跡スケーリングポリシーを `for Redis Auto Scaling ElastiCache` という名前の `for Redis` クラスターに適用します。 `myscalablecluster ElastiCache ElastiCachePrimaryEngineCPUUtilization` 事前定義メトリクスに基づいてポリシー設定を使用します。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
  "PolicyName": "myscalablepolicy",
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:NodeGroups",
  "PolicyType": "TargetTrackingScaling",
```

```
"TargetTrackingScalingPolicyConfiguration": {
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
  {
    "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
  }
}
```

スケーリングポリシーの編集

AWS Management Console、AWS CLI、またはアプリケーションの Auto Scaling API を使用してスケーリングポリシーを編集できます。

AWS Management Console を使用したスケーリングポリシーの編集

ElastiCache for Redis クラスターの Auto Scaling ポリシーを編集するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します (クラスター名の左にあるボタンではなく、クラスター名を選択)。
4. [Auto Scaling ポリシー] タブを選択します。
5. [Scaling policies] (スケーリングポリシー) で、変更する Auto Scaling ポリシーの左にあるボタンを選択して、[Modify] (変更) を選択します。
6. ポリシーに必要な変更を行います。
7. [Modify] (変更) を選択します。

AWS CLI または API を使用したスケーリングポリシーの編集

AWS CLI またはアプリケーションの Auto Scaling API を使用して、スケーリングポリシーを適用するのと同じ方法でスケーリングポリシーを編集できます。

- AWS CLI を使用する場合は、編集するポリシーの名前を `--policy-name` パラメータで指定します。変更するパラメータの新しい値を指定します。
- アプリケーションの Auto Scaling API を使用する場合は、編集するポリシーの名前を `PolicyName` パラメータで指定します。変更するパラメータの新しい値を指定します。

詳細については、「[スケーリングポリシーの適用](#)」を参照してください。

スケーリングポリシーの削除

AWS Management Console、AWS CLI、またはアプリケーションの Auto Scaling API を使用してスケーリングポリシーを削除できます。

AWS Management Console を使用したスケーリングポリシーの削除

ElastiCache for Redis クラスターの Auto Scaling ポリシーを削除するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. Auto Scaling ポリシーを編集するクラスターを選択します (クラスター名の左にあるボタンではなく、クラスター名を選択)。
4. [Auto Scaling ポリシー] タブを選択します。
5. [Scaling policies] (スケーリングポリシー) で、Auto Scaling ポリシーを選択してから [Delete] (削除) を選択します。

AWS CLI を使用したスケーリングポリシーの削除

ElastiCache for Redis クラスターのスケーリングポリシーを削除するには、以下のパラメータを指定して [delete-scaling-policy](#) AWS CLI コマンドを使用します。

- `--policy-name` – スケーリングポリシーの名前。
- `--resource-id` – ElastiCache for Redis のリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、例えば `replication-group/myscalablecluster` です。
- `--service-namespace` – この値は `elasticache` に設定します。
- `--scalable-dimension` – この値は `elasticache:replication-group:NodeGroups` に設定します。

次の例では、`myscalablepolicy` というターゲット追跡スケーリングポリシーを `myscalablecluster` という名前の ElastiCache for Redis クラスターから削除します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling delete-scaling-policy \  
  --policy-name myscalablepolicy \  
  --resource-id replication-group/myscalablecluster \  
  --service-namespace elasticache \  
  --scalable-dimension elasticache:replication-group:NodeGroups
```

Windows の場合:

```
aws application-autoscaling delete-scaling-policy ^  
  --policy-name myscalablepolicy ^  
  --resource-id replication-group/myscalablecluster ^  
  --service-namespace elasticache ^  
  --scalable-dimension elasticache:replication-group:NodeGroups
```

API を使用したスケーリングポリシーの削除

スケーリングポリシーを ElastiCache for Redis クラスターから削除するには、以下のパラメータを指定して [DeleteScalingPolicy](#) AWS CLI コマンドを使用します。

- `--policy-name` – スケーリングポリシーの名前。
- `--resource-id` – ElastiCache for Redis のリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、例えば `replication-group/myscalablecluster` です。
- `--service-namespace` – この値は `elasticache` に設定します。
- `--scalable-dimension` – この値は `elasticache:replication-group:NodeGroups` に設定します。

次の例では、`myscalablepolicy` というターゲット追跡スケーリングポリシーを `myscalablecluster` という名前の ElastiCache for Redis クラスターから削除します。

```
POST / HTTP/1.1  
Host: autoscaling.us-east-2.amazonaws.com  
Accept-Encoding: identity  
Content-Length: 219  
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy  
X-Amz-Date: 20160506T182145Z  
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8  
Content-Type: application/x-amz-json-1.1  
Authorization: AUTHPARAMS
```

```
{
  "PolicyName": "myscalablepolicy",
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:NodeGroups"
}
```

Auto Scaling ポリシーで AWS CloudFormation を使用する

このスニペットでは、ターゲット追跡ポリシーを作成し、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、そのポリシーを [AWS::ElastiCache::ReplicationGroup](#) リソースに適用する方法を示しています。また、[Fn::Join](#) および [Ref](#) 組み込み関数を使用して、同じテンプレートで指定された `AWS::ElastiCache::ReplicationGroup` リソースの論理名で `ResourceId` プロパティを作成します。

```
ScalingTarget:
  Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
  Properties:
    MaxCapacity: 3
    MinCapacity: 1
    ResourceId: !Sub replication-group/${logicalName}
    ScalableDimension: 'elasticache:replication-group:NodeGroups'
    ServiceNamespace: elasticache
    RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

ScalingPolicy:
  Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
  Properties:
    ScalingTargetId: !Ref ScalingTarget
    ServiceNamespace: elasticache
    PolicyName: testpolicy
    PolicyType: TargetTrackingScaling
    ScalableDimension: 'elasticache:replication-group:NodeGroups'
    TargetTrackingScalingPolicyConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ElastiCachePrimaryEngineCPUUtilization
      TargetValue: 40
```

スケジュールされたスケーリング

スケジュールに基づくスケーリングにより、予想可能な需要の変化に応じてアプリケーションを拡張することができます。スケジュールに基づくスケーリングを使用するには、スケジュールされたアクションを作成します。それにより、指定された時間に規模の拡大や縮小を行うように ElastiCache for Redis に伝えます。スケジュールされたアクションを作成する際、既存の ElastiCache for Redis を指定して、スケーリングアクティビティが起こる時刻、最小容量、最大容量を指定できます。スケジュールされたアクションは、1 度だけスケールする、または定期的なスケジュールに従ってスケールするものを作成できます。

既に存在する ElastiCache for Redis 用のスケジュールされたアクションのみを作成できます。スケジュールされたアクションは、クラスターの作成と同時に作成することはできません。

スケジュールされたアクションの作成、管理、削除に関する用語の詳細については、「[スケジュールされたアクションの作成、管理、削除に一般的に使用されるコマンド](#)」を参照してください。

定期的なスケジュールで作成するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します。
4. [アクション] ドロップダウンから [Auto Scaling ポリシーを管理する] を選択します。
5. [Auto Scaling ポリシー] タブを選択します。
6. [Auto Scaling ポリシー] セクションで、[スケーリングポリシーの追加] ダイアログボックスが表示されます。[スケジュールされたスケーリング] を選択します。
7. [Policy Name] では、このポリシー名を入力します。
8. [スケーラブルディメンション] では、[シャード] を選択します。
9. [ターゲットシャード] では、値を選択します。
10. [繰り返し] では、繰り返しを選択します。
11. [頻度]では、それぞれの値を選択します。
12. [開始日] および [開始時間] では、ポリシーが有効になる時刻を選択します。
13. [Add policy] を選択します。

1 回のスケジュールされたアクションを作成するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します。
4. [アクション] ドロップダウンから [Auto Scaling ポリシーを管理する] を選択します。
5. [Auto Scaling ポリシー] タブを選択します。
6. [Auto Scaling ポリシー] セクションで、[スケーリングポリシーの追加] ダイアログボックスが表示されます。[スケジュールされたスケーリング] を選択します。
7. [Policy Name] では、このポリシー名を入力します。
8. [スケーラブルディメンション] では、[シャード] を選択します。
9. [ターゲットシャード] では、値を選択します。
10. [繰り返し] では、[1 回] を選択します。
11. [開始日] および [開始時間] では、ポリシーが有効になる時刻を選択します。
12. 終了日では、ポリシーが有効になるときの日付を選択します。
13. [Add policy] を選択します。

スケジュールされたアクションを削除するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します。
4. [アクション] ドロップダウンから [Auto Scaling ポリシーを管理する] を選択します。
5. [Auto Scaling ポリシー] タブを選択します。
6. [Auto Scaling Policies (Auto Scaling ポリシー)] セクションで Auto Scaling ポリシーを選択してから、[Actions (アクション)] メニューから [Delete (削除)] を選択します。

AWS CLI を使用してスケジュールされたスケーリングを管理するには

次のアプリケーション自動スケーリング API を使用します。

- `put-scheduled-action` <https://docs.aws.amazon.com/cli/latest/reference/autoscaling/put-scheduled-action.html>
- `describe-scheduled-actions` <https://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-scheduled-actions.html>
- `delete-scheduled-action` <https://docs.aws.amazon.com/cli/latest/reference/autoscaling/delete-scheduled-action.html>

AWS CloudFormation を使用して、スケジュールされたアクションを作成するには

このスニペットでは、ターゲット追跡ポリシーを作成

し、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、そのポリシーを [AWS::ElastiCache::ReplicationGroup](#) リソースに適用する方法を示しています。

また、[Fn::Join](#) および [Ref](#) 組み込み関数を使用して、同じテンプレートで指定された `AWS::ElastiCache::ReplicationGroup` リソースの論理名で `ResourceId` プロパティを作成します。

```
ScalingTarget:
  Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
  Properties:
    MaxCapacity: 3
    MinCapacity: 1
    ResourceId: !Sub replication-group/${logicalName}
    ScalableDimension: 'elasticache:replication-group:NodeGroups'
    ServiceNamespace: elasticache
    RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
    ScheduledActions:
      - EndTime: '2020-12-31T12:00:00.000Z'
        ScalableTargetAction:
          MaxCapacity: '5'
          MinCapacity: '2'
          ScheduledActionName: First
          Schedule: 'cron(0 18 * * ? *)'
```

レプリカでの Auto Scaling の使用

以下では、ターゲット追跡とスケジュールされたポリシーの詳細と、AWS Management Console、AWS CLI および API を使用してそれらを適用する方法を説明します。

ターゲット追跡スケーリングポリシー

ターゲット追跡スケーリングポリシーで、メトリクスを選択してターゲット値を設定します。ElastiCache for Redis の AutoScaling は、スケーリングポリシーをトリガーする CloudWatch アラームを作成および管理し、メトリクスとターゲット値に基づいてスケーリング調整値を計算します。スケーリングポリシーは、指定されたターゲット値、またはそれに近い値にメトリクスを維持するため、必要に応じてすべてのシャードで均一にレプリカを追加または削除します。ターゲットの追跡スケーリングポリシーは、メトリクスをターゲット値近くに維持することに加えて、負荷パターンの変動によるメトリクスの変動に合わせて調整し、フリートの容量の急速な変動を最小化します。

レプリカの Auto Scaling 基準

Auto Scaling ポリシーでは、クラスターの次の事前定義されたメトリクスを定義します。

`ElastiCacheReplicaEngineCPUUtilization`: ElastiCache for Redis が auto-scaling オペレーションをトリガーするために使用するすべてのレプリカで集計された AVG EngineCPU 使用率のしきい値。使用率ターゲットは 35 パーセントから 70 パーセントの間で設定できます。

サービスが `ElastiCacheReplicaEngineCPUUtilization` メトリクスが Target 設定以上であることを検出した場合、シャードでレプリカを自動的に増加させます。ElastiCache for Redis は、ターゲットからの変動率と 1 つのレプリカの 2 つの数字のうち、大きい方の数に等しい数だけクラスターレプリカをスケールアウトします。スケールインの場合、全体的なメトリクス値が定義されたターゲットの 75% を下回らない限り、ElastiCache for Redis は自動スケールインしません。

スケールアウトの例では、それぞれ 5 つのシャードと 1 つのレプリカがある場合:

ターゲットが 30% 超えた場合、Redis for ElastiCache はすべてのシャードで 1 つのレプリカ (最大 (0.3、デフォルト 1)) だけスケールアウトします。これにより、それぞれ 2 つのレプリカを持つ 5 つのシャードになります。

スケールインの例で、ターゲット値として 60% を選択した場合、ElastiCache for Redis は、メトリクスが 45% 以下 (ターゲットの 60% を 25% 下回る) になるまで自動スケールインしません。

Auto Scaling に関する考慮事項

次の考慮事項に注意が必要です。

- ターゲットの追跡スケーリングポリシーでは、指定されたメトリクスがターゲット値を超えている場合、スケールアウトする必要があると見なされます。指定されたメトリクスがターゲット値を下回っている場合、ターゲットの追跡スケーリングポリシーを使用してスケールアウトすることでは

きません。ElastiCache for Redis は、クラスター内のすべてのシャードで既存のレプリカの最大値 (Target から切り捨てられた偏差 %、デフォルト 1) でレプリカをスケールアウトします。

- 指定されたメトリクスに十分なデータがない場合、ターゲットの追跡スケールリングポリシーによってスケールされません。不十分なデータは低い使用率として解釈されないため、スケールインされません。
- ターゲット値と実際のメトリクスデータポイント間にギャップが発生する場合があります。これは、ElastiCache for Redis の Auto Scaling が追加または削除する容量を決定するときに、その数を切り上げまたは切り捨てて常に控えめに動作するためです。これにより、不十分な容量を追加したり、必要以上に容量を削除することを防ぎます。
- アプリケーションの可用性を高めるために、サービスのスケールアウトはメトリクスに比例して可能な限り高速に行われますが、スケールインはより緩やかで、クラスター内のシャードで 1 個のレプリカの最大スケールインで行われます。
- それぞれが異なるメトリクスを使用していれば、ElastiCache for Redis クラスターに対して複数のターゲット追跡スケールリングポリシーを設定できます。ElastiCache for Redis の Auto Scaling の目的は常に可用性を優先することであるため、その動作は、ターゲット追跡ポリシーでスケールアウトまたはスケールインの準備ができていないかによって異なります。ターゲット追跡ポリシーのいずれかでスケールアウトする準備ができると、サービスがスケールアウトされますが、すべてのターゲット追跡ポリシー (スケールイン部分が有効) でスケールインする準備ができていない場合のみスケールインされます。
- ターゲットの追跡スケールリングポリシーのために ElastiCache for Redis の Auto Scaling が管理する CloudWatch アラームを編集または削除しないでください。ElastiCache for Redis の Auto Scaling は、スケールリングポリシーを削除するかクラスターを削除するときに、アラームを自動的に削除します。
- ElastiCache for Redis の Auto Scaling は、シャード間でレプリカを手動で変更できなくすることはありません。これらの手動調整は、スケールリングポリシーにアタッチされている CloudWatch アラームに影響しませんが、これらの CloudWatch アラームをトリガーする可能性のあるメトリクスに影響する可能性があります。
- Auto Scaling によって管理されるこれらの CloudWatch アラームは、クラスター内のすべてのシャードでの AVG メトリクスで定義されます。したがって、ホットシャードを持つと、次のいずれかのシナリオが発生する可能性があります。
 - CloudWatch アラームをトリガーするいくつかのホットシャードへの負荷が原因で、必要のない場合にスケールリングする
 - アラームが違反しないように影響を及ぼすすべてのシャードで集約された AVG が原因で、必要な場合にスケールリングしない。

- クラスターごとのノードに対する ElastiCache for Redis のデフォルト制限は引き続き適用されます。したがって、Auto Scaling を選択するとき、最大ノード数がデフォルトの制限を超えると予測される場合は、[\[AWS サービス制限\]](#) で制限の増加をリクエストし、制限タイプ [インスタンスタイプごとのクラスターあたりのノード] を選択します。
- スケールアウト時に必要な、VPC で十分な ENI (Elastic Network Interfaces) が使用可能であることを確認します。詳細については、「[Elastic Network Interface](#)」を参照してください。
- EC2 からの十分な容量がない場合、ElastiCache for Redis Auto Scaling は、容量が利用可能になるまで、または十分な容量を持つインスタンスタイプにクラスターを手動で変更するまで、スケールアウトしません。
- ElastiCache for Redis の Auto Scaling は、25% 下回る ReservedMemoryPercent を持つクラスターがあるレプリカのスケールアップをサポートしません。詳細については、「[リザーブドメモリの管理](#)」を参照してください。

スケールアップポリシーの追加

を使用してスケールアップポリシーを追加できます AWS Management Console。

を使用してスケールアップポリシーを追加します。 AWS Management Console

Redis にauto スケールアップポリシーを追加するには ElastiCache

1. AWS Management Console にサインインし、<https://console.aws.amazon.com/elasticache/> にある Amazon ElastiCache コンソールを開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します (クラスター名の左にあるボタンではなく、クラスター名を選択)。
4. [Auto Scaling ポリシー] タブを選択します。
5. [add dynamic scaling] (動的なスケールアップを追加) を選択します。
6. [Scaling policies] (スケールアップポリシー) の下で、[Add dynamic scaling] (動的なスケールアップを追加) を選択します。
7. [Policy Name] では、このポリシー名を入力します。
8. [スケールアップディメンション] では、ダイアログボックスから [レプリカ] を選択します。
9. 目標値には、ElastiCache レプリカ上で維持したい CPU 使用率の平均値を入力します。この値は、 ≥ 35 かつ ≤ 70 である必要があります。クラスターレプリカが追加または削除され、メトリクスが指定された値に近い値に維持されます。

10. (オプション) スケールインまたはスケールアウトのクールダウン期間は、コンソールからはサポートされていません。AWS CLI を使用してクールダウン値を変更します。
11. [最小容量] には、ElastiCache for Redis Auto Scaling ポリシーで維持する必要があるレプリカの最小数を入力します。
12. [最大容量] には、ElastiCache for Redis Auto Scaling ポリシーで維持する必要があるレプリカの最大数を入力します。この値は、 ≥ 5 である必要があります。
13. [作成] を選択します。

スケーラブルなターゲットの登録

事前定義されたメトリクスまたはカスタムメトリクスに基づいて、スケーリングポリシーを適用できます。そのためには、AWS CLI またはアプリケーションの Auto Scaling API を使用できます。最初のステップは、Redis ElastiCache 用レプリケーショングループを Redis auto ElastiCache スケーリング用に登録することです。

ElastiCache for Redis クラスターで Redis auto スケーリングを使用する前に、クラスターを Redis auto ElastiCache スケーリング用に登録します。ElastiCache そのクラスターに適用されるスケーリングディメンションと制限を定義するために行います。ElastiCache for Redis auto Scaling は、`elasticache:replication-group:Replicas` シャードあたりのクラスターレプリカの数を表すスケーラブルディメンションに沿って ElastiCache for Redis クラスターを動的にスケーリングします。

CLI の使用

ElastiCache クラスターを登録するには、[register-scalable-target](#) 以下のパラメーターを指定してコマンドを実行します。

- `--service-namespace` – この値は `elasticache` に設定します。
- `--resource-id` — クラスターのリソース識別子。ElastiCache このパラメーターの場合、リソースタイプは、たとえば `for Redis ReplicationGroup` クラスターの名前で、一意の識別子は ElastiCache for Redis クラスターの名前です。 `replication-group/myscalablecluster`
- `--scale-dimension` — この値は `elasticache:replication-group:Replicas` に設定します。
- `--min-capacity` — Redis auto ElastiCache スケーリングで管理されるレプリカの最小数です。 `--min-capacity`、`--max-capacity`、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。

- `--max-capacity` — Redis auto ElastiCache スケーリングで管理できるレプリカの最大数です。— `min-capacity`、— `max-capacity`、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。

Example

次の例では、という名前の for Redis クラスターを登録します。ElastiCache `myscalablecluster`この登録は、クラスターが 1 から 5 個のレプリカを持つよう動的にスケーリングされることを示します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling register-scalable-target \  
  --service-namespace elasticache \  
  --resource-id replication-group/myscalablecluster \  
  --scalable-dimension elasticache:replication-group:Replicas \  
  --min-capacity 1 \  
  --max-capacity 5 \  

```

Windows の場合:

```
aws application-autoscaling register-scalable-target ^\  
  --service-namespace elasticache ^\  
  --resource-id replication-group/myscalablecluster ^\  
  --scalable-dimension elasticache:replication-group:Replicas ^\  
  --min-capacity 1 ^\  
  --max-capacity 5 ^\  

```

API の使用

ElastiCache クラスターを登録するには、[register-scalable-target](#)以下のパラメーターを指定してコマンドを実行します。

- `ServiceNamespace` — この値を `elasticache` に設定します。
- `ResourceID` — クラスターのリソース識別子。ElastiCache このパラメーターの場合、リソースタイプは、たとえば `for Redis ReplicationGroup` クラスターの名前で、一意の識別子は `ElastiCache for Redis` クラスターの名前です。 `replication-group/myscalablecluster`
- `ScalableDimension` — この値をに設定します。 `elasticache:replication-group:Replicas`

- **MinCapacity** — Redis auto ElastiCache スケーリングで管理するレプリカの最小数。—min-capacity、—max-capacity、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。
- **MaxCapacity** — Redis auto ElastiCache スケーリングで管理できるレプリカの最大数。—min-capacity、—max-capacity、およびクラスター内のレプリカ数の関係については、「[最小容量と最大容量](#)」を参照してください。

Example

次の例では、アプリケーションの Auto Scaling API `myscalablecluster` で名前を付けた ElastiCache for Redis クラスターを登録します。この登録は、クラスターが 1~5 個のレプリカを持つよう動的にスケールされることを示します。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:Replicas",
  "MinCapacity": 1,
  "MaxCapacity": 5
}
```

スケーリングポリシーの定義

ターゲット追跡スケーリングポリシー設定は、メトリクスとターゲット値が定義されている JSON ブロックで表されます。JSON ブロックとしてスケーリングポリシー設定をテキストファイルに保存できます。このテキストファイルは、AWS CLI または アプリケーションの Auto Scaling API を呼び出す際に使用します。ポリシー設定構文の詳細については、Application Auto Scaling API リファレンスの「[TargetTrackingScalingPolicyConfiguration](#)」を参照してください。

ターゲット追跡スケーリングポリシー設定を定義するには、次のオプションを使用できます。

トピック

- [事前定義メトリクスの使用](#)
- [スケーリングポリシーの編集](#)
- [スケーリングポリシーの削除](#)
- [Auto Scaling ポリシーで AWS CloudFormation を使用する](#)
- [スケジュールされたスケーリング](#)

事前定義メトリクスの使用

ターゲット追跡スケーリングポリシー設定は、メトリクスとターゲット値が定義されている JSON ブロックで表されます。JSON ブロックとしてスケーリングポリシー設定をテキストファイルに保存できます。このテキストファイルは、AWS CLI または アプリケーションの Auto Scaling API を呼び出す際に使用します。ポリシー設定構文の詳細については、Application Auto Scaling API リファレンスの「[TargetTrackingScalingPolicyConfiguration](#)」を参照してください。

ターゲット追跡スケーリングポリシー設定を定義するには、次のオプションを使用できます。

トピック

- [事前定義メトリクスの使用](#)
- [カスタムメトリクスの使用](#)
- [クールダウン期間の使用](#)
- [スケールインアクティビティの無効化](#)
- [ElastiCache for Redis クラスターへのスケーリングポリシーの適用](#)

事前定義メトリクスの使用

定義済みのメトリクスを使用することにより、ElastiCache for Redis の Auto Scaling のターゲット追跡で動作する ElastiCache for Redis クラスターのターゲット追跡スケーリングポリシーを迅速に定義できます。現在、ElastiCache for Redis は、ElastiCache レプリカの Auto Scaling で次の定義済みメトリックスをサポートしています。

ElastiCacheReplicaEngineCPUUtilization — ElastiCache for Redis クラスターのすべてのレプリカでの CloudWatch の EngineCPUUtilization メトリクスの平均値。ElastiCache for Redis クラスターのすべてのレプリカでの CloudWatch の EngineCPUUtilization メトリクスの平均値。集約されたメトリクス値は、必要な ReplicationGroupId とロールのレプリカについて、ElastiCache for Redis ReplicationGroupId, Role の下の CloudWatch にあります。

スケーリングポリシーで事前定義メトリクスを使用するには、スケーリングポリシーのターゲット追跡構成を作成します。この設定は、事前定義メトリクスの `PredefinedMetricSpecification` と、そのメトリクスのターゲット値の `TargetValue` が含まれている必要があります。

カスタムメトリクスの使用

カスタムメトリクスを使用することで、カスタム要件を満たすターゲット追跡スケーリングポリシーを定義できます。スケーリングに比例して変化する ElastiCache for Redis メトリクスに基づいて、カスタムメトリクスを定義することができます。ElastiCache for Redis のすべてのメトリクスがターゲット追跡に使用できるわけではありません。メトリクスは、有効な使用率メトリクスで、インスタンスの使用頻度を示す必要があります。クラスター内のレプリカの数に比例してメトリクスの値を増減する必要があります。この比例的な増減は、メトリクスデータを使用して、比例的にレプリカの数を増減するために必要です。

Example

次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この設定では、カスタムメトリクスにより、`my-db-cluster` という名前のクラスター内のすべてのレプリカでの平均 CPU 使用率 50% に基づいて、ElastiCache for Redis クラスターが調整されます。

```
{
  "TargetValue": 50,
  "CustomizedMetricSpecification":
  {
    "MetricName": "EngineCPUUtilization",
    "Namespace": "AWS/ElastiCache",
    "Dimensions": [
      {
        "Name": "RelicationGroup",
        "Value": "my-db-cluster"
      },
      {
        "Name": "Role",
        "Value": "REPLICA"
      }
    ],
    "Statistic": "Average",
    "Unit": "Percent"
  }
}
```

クールダウン期間の使用

`ScaleOutCooldown` の値を秒単位で指定して、クラスターをスケールアウトするためのクールダウン期間を追加することができます。同様に、`ScaleInCooldown` の値を秒単位で追加して、クラスターをスケールインするためのクールダウン期間を追加することができます。`ScaleInCooldown` と `ScaleOutCooldown` の詳細については、Application Auto Scaling API リファレンスの「[TargetTrackingScalingPolicyConfiguration](#)」を参照してください。

さい。次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この設定では、ElastiCacheReplicaEngineCPUUtilization事前定義メトリクスを使用して、ElastiCache for Redis クラスターのすべてのレプリカでの平均 CPU 使用率 40% に基づいて、そのクラスターが調整されます。この設定では、10 分間のスケールインのクールダウン期間と 5 分間のスケールアウトのクールダウン期間が提供されます。

```
{"TargetValue": 40.0,
  "PredefinedMetricSpecification":
  {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"},
  "ScaleInCooldown": 600,
  "ScaleOutCooldown": 300
}
```

スケールインアクティビティの無効化

スケールインアクティビティを無効にすることにより、ElastiCache for Redis クラスターでターゲット追跡スケーリングポリシー設定がスケーリングされないようにできます。スケールインアクティビティを無効にすると、スケーリングポリシーによってレプリカが削除されることなく、スケーリングポリシーによって必要に応じて追加されます。

DisableScaleIn のブール値を指定して、クラスターのアクティビティのスケールを有効または無効にすることができます。DisableScaleIn の詳細については、Application Auto Scaling API リファレンスの「[TargetTrackingScalingPolicyConfiguration](#)」を参照してください。

Example

次の例では、スケーリングポリシーのターゲット追跡設定について説明します。この設定では、ElastiCacheReplicaEngineCPUUtilization 事前定義メトリクスは、ElastiCache for Redis クラスターのすべてのレプリカでの平均 CPU 使用率 40% に基づいて、そのクラスターを調整します。この設定では、スケーリングポリシーのスケールインアクティビティが無効になります。

```
{"TargetValue": 40.0,
  "PredefinedMetricSpecification":
  {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"},
  "DisableScaleIn": true
}
```

ElastiCache for Redis クラスターへのスケーリングポリシーの適用

クラスターを ElastiCache for Redis の Auto Scaling に登録し、スケーリングポリシーを定義した後、登録されたクラスターにスケーリングポリシーを適用します。ElastiCache for Redis クラスターにスケーリングポリシーを適用する際には、AWS CLI またはアプリケーション Auto Scaling API を使用できます。

AWS CLI の使用

スケーリングポリシーを ElastiCache for Redis クラスターに適用するには、以下のパラメータを指定して [put-scaling-policy](#) コマンドを使用します。

- `--policy-name` – スケーリングポリシーの名前。
- `--policy-type` — この値は `TargetTrackingScaling` に設定します。
- `--resource-id` – ElastiCache for Redis クラスターのリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、たとえば `replication-group/myscalablecluster` です。
- `—service-namespace` – この値は `elasticache` に設定します。
- `—scalle-dimension` — この値は `elasticache:replication-group:Replicas` に設定します。
- `--target-tracking-scaling-policy-configuration` — ElastiCache for Redis クラスターに使用するターゲット追跡スケーリングポリシー設定。

Example

次の例では、`myscalablepolicy` というターゲット追跡スケーリングポリシーを、`myscalablecluster` という名前の ElastiCache for Redis クラスターに ElastiCache for Redis の Auto Scaling を使用して適用します。そのためには、`config.json` という名前のファイルに保存されているポリシー設定を使用します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling put-scaling-policy \  
  --policy-name myscalablepolicy \  
  --policy-type TargetTrackingScaling \  
  --resource-id replication-group/myscalablecluster \  
  --target-tracking-scaling-policy-configuration file://config.json
```

```
--service-namespace elasticache \  
--scalable-dimension elasticache:replication-group:Replicas \  
--target-tracking-scaling-policy-configuration file://config.json
```

```
{"TargetValue": 40.0,  
  "PredefinedMetricSpecification":  
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"  
    },  
  "DisableScaleIn": true  
}
```

Windows の場合:

```
aws application-autoscaling put-scaling-policy ^  
  --policy-name myscalablepolicy ^  
  --policy-type TargetTrackingScaling ^  
  --resource-id replication-group/myscalablecluster ^  
  --service-namespace elasticache ^  
  --scalable-dimension elasticache:replication-group:Replicas ^  
  --target-tracking-scaling-policy-configuration file://config.json
```

API の使用

アプリケーションの Auto Scaling API を使用してスケーリングポリシーを ElastiCache for Redis クラスターに適用するには、以下のパラメータを指定して、[PutScalingPolicy](#) アプリケーションの Auto Scaling API オペレーションを使用します。

- **PolicyName** – スケーリングポリシーの名前。
- **PolicyType** — この値は `TargetTrackingScaling` に設定します。
- **ResourceID** — ElastiCache for Redis クラスターのリソース識別子。このパラメータでは、リソースタイプは `ReplicationGroup` で、一意の識別子は ElastiCache for Redis クラスターの名前、たとえば `replication-group/myscalablecluster` です。
- **ServiceNamespace** – この値は `elasticache` に設定します。
- **ScalableDimension** — この値は `elasticache:replication-group:Replicas` に設定します。
- **TargetTrackingScalingPolicyConfiguration** — Redis for ElastiCache クラスターに使用するターゲット追跡スケーリングポリシー設定。

Example

次の例では、scalablepolicy というターゲット追跡スケーリングポリシーを、myscalablecluster という名前の ElastiCache for Redis クラスターに ElastiCache for Redis の Auto Scaling を使用して適用します。ElastiCacheReplicaEngineCPUUtilization 事前定義メトリクスに基づいてポリシー設定を使用します。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
  "PolicyName": "myscalablepolicy",
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:Replicas",
  "PolicyType": "TargetTrackingScaling",
  "TargetTrackingScalingPolicyConfiguration": {
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    }
  }
}
```

スケーリングポリシーの編集

AWS Management Console、AWS CLI、またはアプリケーションの Auto Scaling API を使用してスケーリングポリシーを編集できます。

AWS Management Console を使用したスケーリングポリシーの編集

AWS Management Console を使用して、事前定義されたメトリクスタイプのポリシーのみを編集できます。

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. ポリシーを追加するクラスターを選択します (クラスター名の左にあるボタンではなく、クラスター名を選択)。
4. [Auto Scaling ポリシー] タブを選択します。
5. [Scaling policies] (スケーリングポリシー) で、変更する Auto Scaling ポリシーの左にあるボタンを選択して、[Modify] (変更) を選択します。
6. ポリシーに必要な変更を行います。
7. [Modify] (変更) を選択します。
8. ポリシーを変更します。
9. [Modify] (変更) を選択します。

AWS CLI または アプリケーションの Auto Scaling API を使用したスケーリングポリシーの編集

AWS CLI または アプリケーションの Auto Scaling API を使用して、スケーリングポリシーを適用するのと同じ方法でスケーリングポリシーを編集できます。

- アプリケーションの Auto Scaling API を使用する場合は、編集するポリシーの名前を PolicyName パラメータで指定します。変更するパラメータの新しい値を指定します。

詳細については、「[ElastiCache for Redis クラスターへのスケーリングポリシーの適用](#)」を参照してください。

スケーリングポリシーの削除

、AWS CLI またはアプリケーション Auto Scaling API を使用してスケーリングポリシーを削除できます。AWS Management Console

を使用してスケーリングポリシーを削除する。AWS Management Console

AWS Management Consoleを使用して、事前定義されたメトリクスタイプのポリシーのみを編集できます。

1. AWS Management Console にサインインし、<https://console.aws.amazon.com/elasticache/> にある Amazon ElastiCache コンソールを開きます。

2. ナビゲーションペインで [Redis] を選択します。
3. Auto Scaling ポリシーを削除するクラスターを選択します。
4. [Auto Scaling ポリシー] タブを選択します。
5. [Scaling policies] (スケーリングポリシー) で、Auto Scaling ポリシーを選択してから [Delete] (削除) を選択します。

AWS CLI またはアプリケーション Auto Scaling API を使用してスケーリングポリシーを削除する

AWS CLI または Application Auto Scaling API を使用して、ElastiCache クラスターからスケーリングポリシーを削除できます。

CLI

ElastiCache for Redis クラスターからスケーリングポリシーを削除するには、[delete-scaling-policy](#) 以下のパラメーターを指定してコマンドを実行します。

- `--policy-name` – スケーリングポリシーの名前。
- `--resource-id` — Redis 用クラスターのリソース識別子。ElastiCache たとえば、このパラメーターのリソースタイプは `ReplicationGroup`、ElastiCache 一意の識別子はクラスターの名前です。 `replication-group/myscalablecluster`
- `--service-namespace` – この値は `elasticache` に設定します。
- `--scalable-dimension` — この値は `elasticache:replication-group:Replicas` に設定します。

Example

次の例では、`myscalablepolicy` というターゲット追跡スケーリングポリシーを `myscalablecluster` という名前の ELC; クラスターから削除します。

Linux、macOS、Unix の場合:

```
aws application-autoscaling delete-scaling-policy \  
  --policy-name myscalablepolicy \  
  --resource-id replication-group/myscalablecluster \  
  --service-namespace elasticache \  
  --scalable-dimension elasticache:replication-group:Replicas \  
  --target-id myscalablepolicy
```

Windows の場合:

```
aws application-autoscaling delete-scaling-policy ^
  --policy-name myscalablepolicy ^
  --resource-id replication-group/myscalablecluster ^
  --service-namespace elasticache ^
  --scalable-dimension elasticache:replication-group:Replicas ^
```

API

ElastiCache for Redis クラスターからスケーリングポリシーを削除するには、[DeleteScalingPolicy](#)以下のパラメーターを指定してアプリケーションの Auto Scaling API オペレーションを使用します。

- **PolicyName** — スケーリングポリシーの名前。
- **ResourceId** — Redis クラスター用のリソース識別子。ElastiCache たとえば、このパラメーターのリソースタイプは `ReplicationGroup`、ElastiCache 一意の識別子はクラスターの名前です。 `replication-group/myscalablecluster`
- **ServiceNamespace** — この値を `elasticache` に設定します。
- **ScalableDimension** — この値をに設定します。 `elasticache:replication-group:Replicas`

次の例では、Application Auto Scaling API `myscalablecluster` を使用して名付けられた ElastiCache for Redis `myscalablepolicy` クラスターから、という名前のターゲットトラッキングスケーリングポリシーを削除します。

```
POST / HTTP/1.1
>>>>>> mainline
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
  "PolicyName": "myscalablepolicy",
  "ServiceNamespace": "elasticache",
  "ResourceId": "replication-group/myscalablecluster",
  "ScalableDimension": "elasticache:replication-group:Replicas"
}
```


Auto Scaling ポリシーで AWS CloudFormation を使用する

このスニペットでは、スケジュールに基づくアクションを作成し、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、そのアクションを [AWS::ElastiCache::ReplicationGroup](#) リソースに適用する方法を示しています。また、[Fn::Join](#) および [Ref](#) 組み込み関数を使用して、同じテンプレートで指定された [AWS::ElastiCache::ReplicationGroup](#) リソースの論理名で ResourceId プロパティを作成します。

```
ScalingTarget:
  Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
  Properties:
    MaxCapacity: 0
    MinCapacity: 0
    ResourceId: !Sub replication-group/${logicalName}
    ScalableDimension: 'elasticache:replication-group:Replicas'
    ServiceNamespace: elasticache
    RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

ScalingPolicy:
  Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
  Properties:
    ScalingTargetId: !Ref ScalingTarget
    ServiceNamespace: elasticache
    PolicyName: testpolicy
    PolicyType: TargetTrackingScaling
    ScalableDimension: 'elasticache:replication-group:Replicas'
    TargetTrackingScalingPolicyConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ElastiCacheReplicaEngineCPUUtilization
      TargetValue: 40
```

スケジュールされたスケーリング

スケジュールに基づくスケーリングにより、予想可能な需要の変化に応じてアプリケーションを拡張することができます。スケジュールに基づくスケーリングを使用するには、スケジュールされたアクションを作成します。それにより、指定された時間に規模の拡大や縮小を行うように ElastiCache for Redis に伝えます。スケジュールされたアクションを作成する際、既存の ElastiCache for Redis を指定して、スケーリングアクティビティが起こる時刻、最小容量、最大容量を指定できます。スケ

スケジュールされたアクションは、1 度だけスケールする、または定期的なスケジュールに従ってスケールするものを作成できます。

既に存在する ElastiCache for Redis 用のスケジュールされたアクションのみを作成できます。スケジュールされたアクションは、クラスターの作成と同時に作成することはできません。

スケジュールされたアクションの作成、管理、削除に関する用語の詳細については、「[スケジュールされたアクションの作成、管理、削除に一般的に使用されるコマンド](#)」を参照してください。

1 回のスケジュールされたアクションを作成するには

シャードのディメンションと同様です。「[スケジュールされたスケーリング](#)」を参照してください。

スケジュールされたアクションを削除するには

シャードのディメンションと同様です。「[スケジュールされたスケーリング](#)」を参照してください。

AWS CLI を使用してスケジュールされたスケーリングを管理するには

次のアプリケーション自動スケーリング API を使用します。

- `put-scheduled-action`<https://docs.aws.amazon.com/cli/latest/reference/application-autoscaling/put-scheduled-action.html>
- `describe-scheduled-actions`<https://docs.aws.amazon.com/cli/latest/reference/application-autoscaling/describe-scheduled-actions.html>
- `delete-scheduled-action`<https://docs.aws.amazon.com/cli/latest/reference/application-autoscaling/delete-scheduled-action.html>

AWS CloudFormation を使用して、Auto Scaling ポリシーを作成する

このスニペットでは、スケジュールに基づくアクションを作成

し、[AWS::ApplicationAutoScaling::ScalableTarget](#) リソースを使用して、そのアクションを [AWS::ElastiCache::ReplicationGroup](#) リソースに適用する方法を示しています。

また、[Fn::Join](#) および [Ref](#) 組み込み関数を使用して、同じテンプレートで指定された

[AWS::ElastiCache::ReplicationGroup](#) リソースの論理名で ResourceId プロパティを作成します。

```
ScalingTarget:
  Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
  Properties:
    MaxCapacity: 0
    MinCapacity: 0
    ResourceId: !Sub replication-group/${logicalName}
    ScalableDimension: 'elasticache:replication-group:Replicas'
    ServiceNamespace: elasticache
    RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
  ScheduledActions:
    - EndTime: '2020-12-31T12:00:00.000Z'
      ScalableTargetAction:
        MaxCapacity: '5'
        MinCapacity: '2'
        ScheduledActionName: First
        Schedule: 'cron(0 18 * * ? *)'
```

クラスターモードの変更

Redis は、シャーディングとレプリケーションをサポートする分散型インメモリデータベースです。ElastiCache for Redis クラスターは、複数の Redis ノード間でデータを分割できる Redis の分散実装です。ElastiCache for Redis クラスターには、クラスターモード有効 (CME) とクラスターモード無効 (CMD) の 2 つのオペレーションモードがあります。CME では Redis は複数のシャードとノードを備えた分散データベースとして機能しますが、CMD では Redis は単一ノードとして機能します。

CMD から CME に移行するには、次の条件を満たしている必要があります。

Important

クラスターモードの設定は、クラスターモード無効からクラスターモード有効にのみ変更できます。この設定を元に戻すことはできません。

- クラスターのキーはデータベース 0 にのみ存在できます。
- アプリケーションは、クラスタープロトコルを使用できる Redis クライアントを使用し、設定エンドポイントを使用する必要があります。

- 少なくとも 1 つのレプリカがあるクラスターでは、自動フェイルオーバーを有効にする必要があります。
- 移行に必要な Redis エンジンの最小バージョンは 7.0 です。

CMD から CME に移行するには、クラスターモードの設定を、クラスターモード無効からクラスターモード有効に変更する必要があります。これは、移行プロセス中にクラスターの可用性を確保するための 2 段階の手順です。

Note

クラスター有効の設定を持つパラメータグループを指定する必要があります。つまり、クラスター有効パラメータを [yes] に設定します。デフォルトのパラメータグループを使用している場合、ElastiCache for Redis は、クラスター有効の設定になっている対応するデフォルトパラメータグループを自動的に選択します。CMD クラスターでは、クラスター有効パラメータ値が [no] に設定されます。クラスターが互換モードに移行すると、変更アクションの一部としてクラスター有効のパラメータ値が [yes] に更新されます。詳細については、「[パラメータグループを使用したエンジンパラメータの設定](#)」を参照してください。

1. 準備 — テスト用の CME クラスターを作成し、スタックがそのクラスターで動作する準備ができていることを確認します。ElastiCache for Redis には、準備が整っていることを確認する方法はありません。詳細については、「[クラスターの作成](#)」を参照してください。
2. 既存の CMD クラスター設定をクラスターモード互換に変更 – このモードでは、1 つのシャードがデプロイされ、ElastiCache for Redis は単一ノードとしてだけでなく、単一シャードクラスターとしても機能します。互換モードとは、クライアントアプリケーションがどちらかのプロトコルを使用してクラスターと通信できることを意味します。このモードでは、Redis クラスタープロトコルと設定エンドポイントの使用を開始するようにアプリケーションを再設定する必要があります。Redis クラスターモードをクラスターモード互換に変更するには、以下の手順に従います。

Note

互換モードでは、スケーリングやエンジンバージョンなどの他の変更オペレーションはクラスターで実行できません。さらに、[ModifyReplicationGroup](#) リクエスト内でクラス

ターモードのパラメータを定義する場合、パラメータ (cacheParameterGroupName を除く) を変更することはできません。

- a. AWS Management Consoleを使用し、「[レプリケーショングループの変更](#)」を参照してクラスターモードを [互換] に設定します。
- b. API を使用する場合は、「[ModifyReplicationGroup](#)」を参照し、ClusterMode パラメータを [compatible] に更新します。
- c. AWS CLI を使用する場合は、「[modify-replication-group](#)」を参照し、cluster-mode パラメータを [compatible] に更新します。

Redis クラスターモードをクラスターモード互換に変更すると、[DescribeReplicationGroups](#) API は ElastiCache for Redis クラスター設定エンドポイントを返します。クラスター設定エンドポイントは、アプリケーションがクラスターに接続するために使用できる単一のエンドポイントです。詳細については、「[接続エンドポイントの検索](#)」を参照してください。

3. クラスター設定をクラスターモード有効に変更 – クラスターモードをクラスターモード互換に設定したら、次のステップは、クラスター設定をクラスターモード有効に変更します。このモードでは、単一のシャードが実行され、お客様はクラスターをスケーリングしたり、他のクラスター設定を変更したりできます。

クラスターモードを有効に変更するには、次の手順に従います。

始める前に、Redis クライアントがクラスタープロトコルを使用するように移行していることと、およびクラスターの設定エンドポイントが使用中でないことを確認します。

- a. AWS Management Consoleを使用し、「[レプリケーショングループの変更](#)」を参照してクラスターモードを [有効] に設定します。
- b. API を使用する場合は、「[ModifyReplicationGroup](#)」を参照し、ClusterMode パラメータを [enabled] に更新します。
- c. AWS CLI を使用する場合は、「[modify-replication-group](#)」を参照し、cluster-mode パラメータを [enabled] に更新します。

クラスターモードを有効に変更すると、エンドポイントは Redis クラスター仕様に従って設定されます。[DescribeReplicationGroups](#) API は、クラスターモードパラメータを [enabled] として返し、アプリケーションがクラスターに接続するために使用できるようになったクラスターエンドポイントを返します。

クラスターモードを有効に変更すると、クラスターエンドポイントが変更されることに注意してください。必ず、新しいエンドポイントを使用してアプリケーションを更新してください。

また、クラスターモード互換からクラスターモード無効 (CMD) に戻して、元の設定を維持することもできます。

クラスター設定をクラスターモード互換からクラスターモード無効に変更する

1. AWS Management Consoleを使用し、「[レプリケーショングループの変更](#)」を参照してクラスターモードを [無効] に設定します。
2. API を使用する場合は、「[ModifyReplicationGroup](#)」を参照し、ClusterMode パラメータを [disabled] に更新します。
3. AWS CLI を使用する場合は、「[modify-replication-group](#)」を参照し、cluster-mode パラメータを [disabled] に更新します。

クラスターモードを無効に変更すると、[DescribeReplicationGroups](#) API はクラスターモードのパラメータを [disabled] として返します。

グローバルデータストアを使用した AWS リージョン間のレプリケーション

Note

グローバルデータストアは現在、独自設計型クラスターでのみ使用できます。

Global Datastore for Redis 機能を使用すると、AWS リージョン間でフルマネージド、高速、信頼性、安全なレプリケーションを操作できます。この機能を使用すると、ElastiCache for Redis のクロスリージョンリードレプリカクラスターを作成して、AWS リージョン間で低レイテンシーの読み取りとディザスタリカバリを有効にできます。

次のセクションでは、Global Datastore の操作方法について説明します。

トピック

- [概要](#)
- [前提条件と制限](#)

- [Global Datastore の使用 \(コンソール\)](#)
- [Global Datastore の使用 \(CLI\)](#)

概要

各 Global Datastore は、互いにレプリケートする 1 つ以上のクラスターの集合です。

Global datastore は、次のもので構成されます。

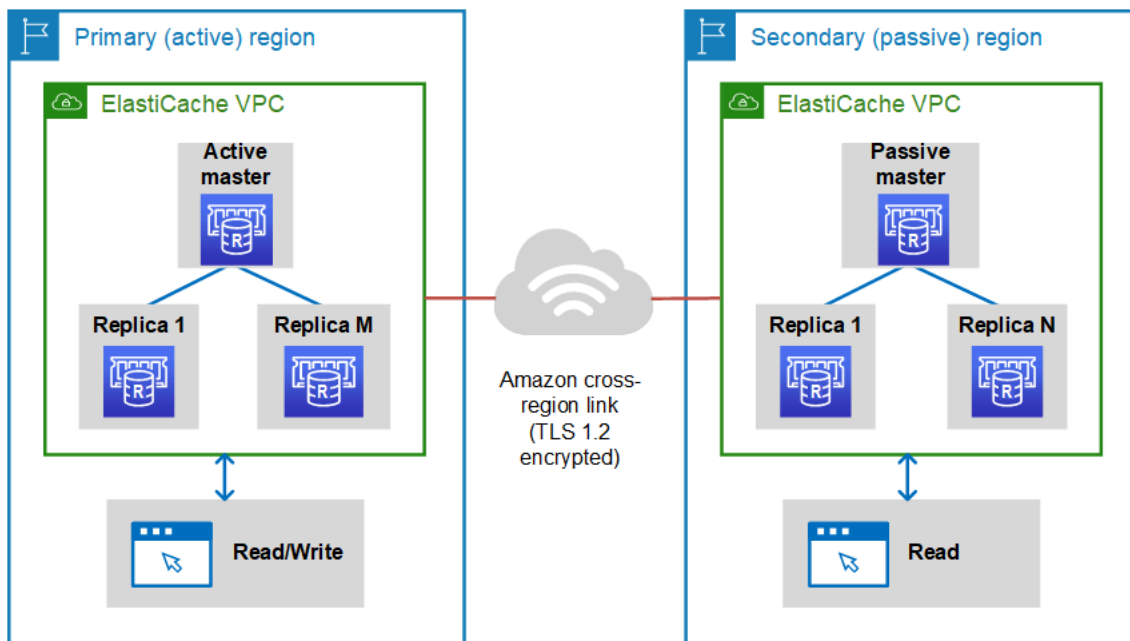
- [プライマリ (アクティブ) クラスター] – プライマリクラスターは、Global Datastore 内のすべてのクラスターにレプリケートされる書き込みを受け入れます。プライマリクラスターは、読み込みリクエストも受け付けます。
- [セカンダリ (パッシブ) クラスター] – セカンダリクラスターは、読み取りリクエストのみを受け入れ、プライマリクラスターからのデータ更新をレプリケートします。セカンダリクラスターは、プライマリクラスターとは異なる AWS リージョンに存在する必要があります。

でグローバルデータストアを作成すると ElastiCache、ElastiCache For Redis はプライマリクラスターからセカンダリクラスターにデータを自動的にレプリケートします。Redis データをレプリケートする AWS リージョンを選択し、その AWS リージョンにセカンダリクラスターを作成します。ElastiCache 次に、は 2 つのクラスター間のデータの自動非同期レプリケーションを設定および管理します。

Redis に Global Datastore を使用すると、次の利点があります。

- 地理的パフォーマンス – 追加の AWS リージョンにリモートレプリカクラスターを設定し、それら間でデータを同期することで、その AWS リージョンのデータアクセスのレイテンシーを短縮できます。グローバルデータストアは、AWS リージョン間で低レイテンシーの地理ローカル読み取りを提供することで、アプリケーションの応答性を高めるのに役立ちます。
- [災害対策] – Global Datastore 内のプライマリクラスターでパフォーマンスが低下した場合は、セカンダリクラスターを新しいプライマリクラスターとして昇格させることができます。これを行うには、セカンダリクラスターを含む任意の AWS リージョンに接続します。

次の図は、Global Datastore がどのように機能するかを示しています。



前提条件と制限

Global Datastore を開始する前に、次の点に注意してください。

- グローバルデータストアは、AWS アジアパシフィック (ソウル、東京、シンガポール、シドニー、ムンバイ、大阪)、欧州 (フランクフルト、パリ、ロンドン、アイルランド、ストックホルム)、米国東部 (バージニア北部およびオハイオ)、米国西部 (北カリフォルニアおよびオレゴン)、南米 (サンパウロ) AWS GovCloud、(米国西部および米国東部)、カナダ (中部) リージョン、中国 (北京および寧夏) の各リージョンでサポートされています。
- グローバルデータストア内のすべてのクラスター (プライマリとセカンダリ) は、プライマリノードの数、ノードタイプ、エンジンバージョン、およびシャードの数が必要があります (クラスターモードが有効な場合)。Global Datastore 内の各クラスターには、そのクラスターのローカルな読み取りトラフィックに対応するために、異なる数のリードレプリカを設定できます。

既存の単一ノードクラスターを使用する場合は、レプリケーションを有効にする必要があります。

- グローバルデータストアは、m5 または r5 より古いインスタンスではサポートされていません。
- プライマリクラスターのレプリケーションは、1 つの AWS リージョンから他の最大 2 つの AWS リージョンのセカンダリクラスターに設定できます。

Note

この例外は、中国 (北京) リージョンと中国 (寧夏) リージョンであり、レプリケーションは2つのリージョン間でしか発生しません。

- Global Datastore は VPC クラスターのみで操作できます。詳細については、「[Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)」を参照してください。EC2-Classic を使用する場合、Global Datastore はサポートされません。詳細については、「[Amazon EC2-Classic](#)」を参照してください。Amazon EC2

Note

現時点では、[ElastiCache での Local Zones の使用](#) でグローバルデータストアを使用することはできません。

- ElastiCache は、あるリージョンから別の AWS リージョンへの自動フェイルオーバーをサポートしていません。必要に応じて、セカンダリクラスターを手動で昇格できます。例については、[セカンダリクラスターのプライマリへの昇格](#)を参照してください。
- 既存のデータからブートストラップするには、既存のクラスターをプライマリとして使用して、Global Datastore を作成します。既存のクラスターをセカンダリとして追加することはできません。クラスターをセカンダリとして追加するプロセスでは、データが消去されるため、データが失われる可能性があります。
- Global Datastore に属するクラスターのローカルパラメータグループを変更すると、パラメータ更新がすべてのクラスターに適用されます。
- リージョンクラスターは、垂直方向 (スケールアップとスケールダウン) と水平方向 (スケールインとスケールアウト) の両方でスケールできます。Global Datastore を変更することで、クラスターを拡張できます。Global Datastore 内のすべてのリージョンクラスターは、中断することなく拡張されます。詳細については、「[Redis ElastiCache のスケーリング](#)」を参照してください。
- Global Datastore は、[保管時の暗号化](#)、[転送中の暗号化](#)、および [Redis AUTH](#) をサポートします。
- グローバルデータストアは、インターネットプロトコルバージョン 6 (IPv6) をサポートしていません。
- グローバルデータストアは AWS KMS キーをサポートします。詳細については、AWS Key Management Service デベロッパーガイドの「[AWS キー管理サービスの概念](#)」を参照してください。

Note

Global Datastore では、次の条件で [pub/sub メッセージング](#) がサポートされます。

- クラスターモードが無効な場合、pub/sub は完全にサポートされます。プライマリ AWS リージョンのプライマリクラスターで発行されたイベントは、セカンダリ AWS リージョンに伝達されます。
- クラスターモードが有効の場合、次のことが適用されます。
 - キースペースにない公開イベントの場合、同じ AWS リージョンのサブスクライバーのみがイベントを受信します。
 - 公開されたキースペースイベントの場合、すべての AWS リージョンのサブスクライバーはイベントを受け取ります。

Global Datastore の使用 (コンソール)

コンソールを使用して Global Datastore を作成するには、次の 2 つのステップから成るプロセスに従います。

1. 既存のクラスターを使用するか、新しいクラスターを作成して、プライマリクラスターを作成します。エンジンは Redis 5.0.6 以降である必要があります。
2. Redis 5.0.6 エンジン以降を使用して、異なる AWS リージョンに最大 2 つのセカンダリクラスターを追加します。

次の手順では、for Redis コンソールを使用して ElastiCache Redis のグローバルデータストアを作成し、その他の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

- [既存のクラスターを使用した Global Datastore の作成](#)
- [新しいプライマリクラスターを使用した新しい Global Datastore の作成](#)
- [Global Datastore 詳細の表示](#)
- [Global Datastore へのリージョンの追加](#)
- [Global Datastore の変更](#)
- [セカンダリクラスターのプライマリへの昇格](#)
- [Global Datastore からのリージョンの削除](#)

• Global Datastore の削除

既存のクラスターを使用した Global Datastore の作成

このシナリオでは、既存のクラスターを使用して、新しい Global Datastore のプライマリとして機能させます。次に、別の AWS リージョンに読み取り専用セカンダリクラスターを作成します。このセカンダリクラスターは、プライマリクラスターから自動更新と非同期更新を受け取ります。

Important

既存のクラスターでは、Redis 5.0.6 以降のエンジンを使用する必要があります。

既存のクラスターを使用して Global Datastore を作成するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択し、グローバルデータストアの作成 を選択します。
3. プライマリクラスター設定ページで、次の操作を行います。
 - グローバルデータストア情報フィールドに、新しいグローバルデータストアの名前を入力します。
 - (省略可能) [説明] の値を入力します。
4. リージョンクラスター で、既存のリージョンクラスター を使用する を選択します。
5. 既存のクラスター で、使用する既存のクラスターを選択します。
6. 次のオプションをそのまま使用します。これらは、プライマリクラスターの設定に合わせて事前に設定されていて、変更することはできません。
 - エンジンバージョン
 - ノードの種類
 - パラメータグループ

Note

ElastiCache は、指定されたパラメータグループの値から新しいパラメータグループを自動生成し、新しいパラメータグループをクラスターに適用します。この新しい

パラメータグループを使用して、Global Datastore のパラメータを変更します。自動生成された各パラメータグループは、1 つの唯一のクラスターにのみ関連付けられます。したがって、1 つの唯一の Global Datastore にのみ関連付けられます。

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

Note

カスタマーマネージド AWS KMS キーを選択し、そのキーを選択することで、別の暗号化キーを指定できます。詳細については、「[カスタマーマネージド AWS KMS キーの使用](#)」を参照してください。

- 転送中の暗号化 – 転送中のデータの暗号化を有効にします。詳細については、「[転送中の暗号化](#)」を参照してください。Redis エンジンバージョン 6.0 以降では、転送中の暗号化を有効にすると、次のアクセスコントロールオプションのいずれかを指定するよう求められます。
 - [アクセスコントロールなし] – これがデフォルトの設定です。これは、制限がないことを示します。
 - [ユーザーグループアクセスコントロールリスト] – ユーザーと使用可能なオペレーションに対する許可のセットを持つユーザーグループを選択します。詳細については、「[コンソールおよび CLI を使用したユーザーグループの管理](#)」を参照してください。
 - [Redis AUTH デフォルトユーザー] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。
7. (オプション) 必要に応じて、残りのセカンダリクラスター設定を更新します。プライマリクラスターと同じ値が事前に入力されていますが、そのクラスターの特定の要件を満たすように更新できます。
- [ポート]
 - レプリケーション数
 - サブネットグループ
 - 優先アベイラビリティーゾーン
 - セキュリティグループ
 - カスタマーマネージド (AWS KMS キー)
 - Redis AUTH トークン

- 自動バックアップの有効化
 - バックアップの保存期間
 - バックアップウィンドウ
 - メンテナンスウィンドウ
 - SNS 通知のトピック
8. [作成] を選択します。これにより、Global Datastore のステータスが [Creating] に設定されます。プライマリクラスターがグローバルデータストアに関連付けられ、セカンダリクラスターが [関連付け] ステータスになった後、ステータスは [変更中] に移行します。

プライマリクラスターとセカンダリクラスターを Global Datastore に関連付けると、ステータスが [Available] に変わります。この時点で、読み取りと書き込みを受け入れるプライマリクラスターと、プライマリクラスターからレプリケートされた読み取りを受け入れるセカンダリクラスターがあります。

Redis ページが更新され、以下を含めて、クラスターが Global Datastore の一部であるかどうかを示されます。


- [Global Datastore] – クラスターが属する Global Datastore の名前。
- [Global Datastore Role] – クラスターのロール (プライマリまたはセカンダリ)。

別の AWS リージョンに最大 1 つのセカンダリクラスターを追加できます。詳細については、「[Global Datastore へのリージョンの追加](#)」を参照してください。

新しいプライマリクラスターを使用した新しい Global Datastore の作成

新しいクラスターがある Global Datastore を作成する場合は、次の手順を実行します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択し、グローバルデータストアの作成 を選択します。
3. [Primary cluster settings] (プライマリクラスターの設定) で、次の作業を行います。
 - a. [Cluster mode] (クラスターモード) で、[Enabled] (有効) または [Disabled] (無効) を選択します。

- b. グローバルデータストア情報には、名前 の値を入力します。はサフィックス ElastiCache を使用して、グローバルデータストアの一意の名前を生成します。ここで指定するサフィックスを使用して、Global Datastore を検索できます。
 - c. (オプション) [Global Datastore Description] に値を入力します。
4. [Regional cluster] (リージョンクラスター) で、次の作業を行います。
 - a. リージョン で、使用可能な AWS リージョンを選択します。
 - b. [Create new regional cluster] (新しいリージョンクラスターを作成) または [Use existing regional cluster] (既存のリージョンクラスターを使用) を選択します。
 - c. [Create new regional cluster] (新しいリージョンクラスターを作成) を選択した場合は、[Cluster info] (クラスター情報) で、クラスターの名前と説明 (オプション) を入力します。
 - d. [Location] (場所) で、[Multi-AZ] (マルチ AZ) および [Auto-failover] (自動フェイルオーバー) のデフォルト設定を受け入れることをお勧めします。
 5. [Cluster settings] (クラスター設定)
 - a. [Engine version] (エンジンバージョン) で、使用可能なバージョン (5.0.6 以降) を選択します。
 - b. [Port] (ポート) で、デフォルトポート 6379 を使用します。異なるポートを使用する理由がある場合は、そのポート番号を入力します。
 - c. [パラメータグループ] で、パラメータグループを選択するか、新しいパラメータグループを作成します。パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「[Redis 固有のパラメータ](#)」および「[パラメータグループを作成する](#)」を参照してください。
-  Note
- パラメータグループを選択してエンジン設定値を設定すると、そのパラメータグループが Global Datastore 内のすべてのクラスターに適用されます。[パラメータグループ] ページの yes/no [グローバル] 属性は、パラメータグループがグローバルデータストアの一部であるかどうかを示します。
- d. [ノードタイプ] で、下向き矢印 (▼) を選択します。[ノードタイプの変更] ダイアログボックスで、必要なノードタイプの [イン

スタンスファミリー] の値を選択します。次に、このクラスターで使用するノードタイプを選択し、[保存] を選択します。

詳細については、「[ノードサイズを選択](#)」を参照してください。

r6gd ノードタイプを選択すると、データ階層化が自動的に有効になります。詳細については、「[データ階層化](#)」を参照してください。

e. Redis (クラスターモードが無効) クラスターを作成する場合:

[Number of replicas] (レプリケーション数) で、このクラスターに必要なレプリケーションの数を選択します。

f. Redis (クラスターモードが有効) クラスターを作成する場合:

i. [シャード数] で、この Redis (クラスターモードが有効) クラスターに必要なシャード (パーティション/ノードグループ) の数を選択します。

Redis (クラスターモードが有効) の一部のバージョンでは、クラスター内のシャード数を動的に変更できます。

- [Redis 3.2.10 以降] – クラスターで Redis 3.2.10 以降のバージョンを実行している場合は、クラスター内のシャード数を動的に変更できます。詳細については、「[Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)」を参照してください。

- [その他の Redis バージョン] – クラスターでバージョン 3.2.10 より前のバージョンの Redis を実行している場合は、別の方法があります。この場合、クラスター内のシャード数を変更するには、新しいシャード数を使用して新しいクラスターを作成します。詳細については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

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

Redis (クラスターモードが有効) には、次の制限があります。

- マルチ AZ が有効になっている場合は、シャードごとに少なくとも 1 つのレプリカがあることを確認してください。
- コンソールを使用してクラスターを作成する場合、シャードごとのレプリカ数は同じになります。
- シャードあたりのリードレプリカ数は固定され、変更できません。シャード (API/CLI: ノードグループ) あたりのレプリカ数を増減する必要がある場合は、新しいレプ

リカ数で新しいクラスターを作成する必要があります。詳細については、「[外部で作成されたバックアップによる新しい独自設計型クラスターのシード](#)」を参照してください。

- サブネットグループ設定で、このクラスターに適用するサブネットを選択します。はデフォルトの IPv4 サブネットグループ ElastiCache を提供するか、新しい IPv4 サブネットグループを作成するかを選択できます。IPv6 の場合は、IPv6 CIDR ブロックを使用するサブネットグループを作成する必要があります。デュアルスタックを選択した場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

詳細については、「[VPC にサブネットを作成する](#)」を参照してください。

- [Availability zone placements] (アベイラビリティゾーンの配置) には 2 つのオプションがあります。

- 設定なし — アベイラビリティ ElastiCache ゾーンを選択します。
- [アベイラビリティゾーンの指定] – 各クラスターに対するアベイラビリティゾーンを指定します。

アベイラビリティゾーンの指定を選択した場合、クラスターのシャードごとにリストからアベイラビリティゾーンを選択します。


詳細については、「[リージョンとアベイラビリティゾーンの選択](#)」を参照してください。

The screenshot shows the configuration interface for an Amazon ElastiCache cluster. It features two dropdown menus: 'Slots and keyspaces' set to 'Custom distribution' and 'Availability zone(s)' set to 'Specify availability zones'. Below these is a table with three columns: 'NodeGroup', 'Slots/Keyspaces', 'Primary', and 'Replica 1'. The 'NodeGroup' column lists 'NodeGroup 1', 'NodeGroup 2', and 'NodeGroup 3'. The 'Slots/Keyspaces' column contains '0-1234', an empty field, and another empty field. The 'Primary' column shows 'us-east-1a', 'us-east-1b', and 'us-east-1a'. The 'Replica 1' column shows 'us-east-1a', 'us-east-1a', and 'us-east-1a'. Red circles highlight the '0-1234' value and the 'us-east-1b' dropdown selection.

NodeGroup	Slots/Keyspaces	Primary	Replica 1
NodeGroup 1	0-1234	us-east-1a	us-east-1a
NodeGroup 2		us-east-1b	us-east-1a
NodeGroup 3		us-east-1a	us-east-1a


キースペースとアベイラビリティゾーンの指定

8. [Next] (次へ) を選択します。
9. [Advanced Redis settings] (Redis の詳細設定) で
 - [Security] (セキュリティ):
 - i. データを暗号化するには、次のオプションがあります。
 - 保管時の暗号化 – ディスクに保存されているデータの暗号化を有効にします。詳細については、「[保管時の暗号化](#)」を参照してください。

 Note

カスタマーマネージド AWS KMS キーを選択し、そのキーを選択することで、別の暗号化キーを指定できます。詳細については、「[AWS KMS のカスタマー管理の CMK の使用](#)」を参照してください。

- [転送中の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「[転送中の暗号化](#)」を参照してください。Redis エンジンバージョン 6.0 以降では、転送中の暗号化を有効にすると、次のアクセスコントロールオプションのいずれかを指定するよう求められます。
 - アクセスコントロールなし – これがデフォルトの設定です。これは、クラスターへのユーザーアクセスに制限がないことを示します。
 - [ユーザーグループのアクセスコントロールリスト] – クラスターにアクセスできるユーザーのセットが定義されているユーザーグループを選択します。詳細については、「[コンソールおよび CLI を使用したユーザーグループの管理](#)」を参照してください。
 - [Redis AUTH デフォルトユーザー] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。
- [Redis AUTH] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。

 Note

3.2.6 以降の Redis バージョン (バージョン 3.2.10 を除く) では、Redis AUTH のみがオプションとなります。

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

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

10. 自動バックアップを定期的にスケジュールする場合は、[自動バックアップの有効化] を選択し、自動バックアップを保持して自動的に削除するまでの日数を入力します。自動バックアップを定期的にスケジュールしない場合は、[自動バックアップを有効化] チェックボックスをオフにします。いずれの場合も、常に手動バックアップを作成するオプションがあります。

Redis のバックアップと復元の詳細については、「[スナップショットおよび復元](#)」を参照してください。

11. (オプション) メンテナンスウィンドウを指定します。メンテナンスウィンドウは、がクラスターのシステムメンテナンスを ElastiCache スケジュールする毎週の、通常は 1 時間の長さです。ElastiCache でメンテナンスウィンドウの日時を選択 (設定なし) することも、日、時刻、期間を自分で選択 (メンテナンスウィンドウを指定) することもできます。メンテナンスウィンドウを指定を選択した場合は、リストからメンテナンス期間の Start day、開始時間および期間を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「[メンテナンスの管理](#)」を参照してください。


12. (オプション) [ログ]:

- [ログの形式] の下で、[テキスト] または [JSON] を選択します。
- 送信先タイプで、CloudWatch ログ または Kinesis Firehose を選択します。
- ログ送信先で、新規作成を選択して CloudWatch ログロググループ名または Firehose ストリーム名を入力するか、既存の選択を選択して CloudWatch ログロググループ名または Firehose ストリーム名を選択します。

13. タグでは、クラスターやその他の ElastiCache リソースの管理に役立つように、タグ形式で各リソースに独自のメタデータを割り当てることができます。詳細については、「[ElastiCache リソースのタグ付け](#)」を参照してください。
14. すべてのエントリと選択を確認し、必要な修正を行います。準備ができたら、[次へ] を選択します。
15. 前のステップでクラスターを設定したら、セカンダリクラスターの詳細を設定します。
16. リージョンクラスターで、クラスターが配置されている AWS リージョンを選択します。


17. [Cluster info] (クラスター情報) で、クラスターの名前と説明 (オプション) を入力します。
18. 次のオプションは、プライマリクラスター設定と一致するように事前入力されていて、変更できません。

- ロケーション
- エンジンバージョン
- インスタンスタイプ
- ノードの種類
- シャード数
- パラメータグループ

 Note

ElastiCache は、指定されたパラメータグループの値から新しいパラメータグループを自動生成し、新しいパラメータグループをクラスターに適用します。この新しいパラメータグループを使用して、Global Datastore のパラメータを変更します。自動生成された各パラメータグループは、1 つの唯一のクラスターにのみ関連付けられます。したがって、1 つの唯一の Global Datastore にのみ関連付けられます。


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

 Note

カスタマーマネージド AWS KMS キーを選択し、そのキーを選択することで、別の暗号化キーを指定できます。詳細については、「[カスタマーマネージド AWS KMS キーの使用](#)」を参照してください。

- 転送中の暗号化 – 転送中のデータの暗号化を有効にします。詳細については、「[転送中の暗号化](#)」を参照してください。Redis エンジンバージョン 6.4 以上では、転送中の暗号化を有効にすると、次の [Access Control] (アクセスコントロール) オプションのいずれかを指定するよう求められます。
 - アクセスコントロールなし – これがデフォルトの設定です。これは、クラスターへのユーザーアクセスに制限がないことを示します。

- [ユーザーグループのアクセスコントロールリスト] — クラスターにアクセスできるユーザーセットが定義されているユーザーグループを選択します。詳細については、「[コンソールおよび CLI を使用したユーザーグループの管理](#)」を参照してください。
- [Redis AUTH デフォルトユーザー] – Redis サーバーの認証メカニズムです。詳細については、「[Redis AUTH](#)」を参照してください。

 Note

Redis のバージョン 4.0.2 の間では、転送時の暗号化が最初にサポートされましたが、バージョン 6.0.4 では、Redis AUTH が唯一のオプションです。

セカンダリクラスターの残りの設定には、プライマリクラスターと同じ値が事前に入力されますが、そのクラスターの特定の要件を満たすように以下の項目を更新できます。

- [ポート]
 - レプリケーション数
 - サブネットグループ
 - 優先アベイラビリティーゾーン
 - セキュリティグループ
 - カスタマーマネージド (AWS KMS キー)
 - Redis AUTH トークン
 - 自動バックアップの有効化
 - バックアップの保存期間
 - バックアップウィンドウ
 - メンテナンスウィンドウ
 - SNS 通知のトピック
19. [作成] を選択します。これにより、Global Datastore のステータスが [Creating] に設定されます。プライマリクラスターとセカンダリクラスターを Global Datastore に関連付けると、ステータスが [Available] に変わります。読み取りと書き込みを受け入れるプライマリクラスターと、プライマリクラスターからレプリケートされた読み取りを受け入れるセカンダリクラスターを用意しました。

Redis ページも更新され、以下を含めて、クラスターが Global Datastore の一部であるかどうかを示されます。

- [Global Datastore] – クラスターが属する Global Datastore の名前。
- [Global Datastore Role] – クラスターのロール (プライマリまたはセカンダリ)。

別の AWS リージョンに最大 1 つのセカンダリクラスターを追加できます。詳細については、「[Global Datastore へのリージョンの追加](#)」を参照してください。

Global Datastore 詳細の表示

既存のグローバルデータストアの詳細を表示したり、グローバルデータストアページで変更したりできます。

Global Datastore の詳細を表示するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストアを選択し、使用可能なグローバルデータストアを選択します。

続いて、次の Global Datastore プロパティを調べることができます。

- Global Datastore 名: Global Datastore の名前
- 説明: Global Datastore の説明
- ステータス: 次のオプションがあります。
 - [作成中]
 - 変更中
 - 利用可能
 - [削除中]
 - [Primary-Only] - このステータスは、Global Datastore にプライマリクラスターのみが含まれていることを示します。すべてのセカンダリクラスターが削除されるか、正常に作成されません。
- クラスターモード: 有効または無効のいずれか
- Redis エンジンバージョン: Global Datastore を実行する Redis エンジンバージョン
- インスタンスノードタイプ: Global Datastore に使用されるノードタイプ

- 保存時の暗号化: 有効または無効のいずれか
- 転送時の暗号化: 有効または無効のいずれか
- Redis AUTH: 有効または無効のいずれか

Global Datastore に対して次の変更を行うことができます。

- [Global Datastore へのリージョンの追加](#)
- [Global Datastore からのリージョンの削除](#)
- [セカンダリクラスターのプライマリへの昇格](#)
- [Global Datastore の変更](#)

[Global Datastore] ページには、Global Datastore を構成する個々のクラスターと、それぞれの次のプロパティも一覧表示されます。

- リージョン - クラスターが保存されている AWS リージョン
- [Role] - プライマリまたはセカンダリのいずれか
- [Cluster name] - クラスターの名前
- [Status] - 次のオプションがあります。
 - [Associating] - クラスターを Global Datastore に関連付けています。
 - [Associated] - クラスターは Global Datastore に関連付けられています。
 - [Disassociating] - Global Datastore 名を使用して、Global Datastore からセカンダリクラスターを削除するプロセス。その後、セカンダリクラスターはプライマリクラスターから更新を受信しなくなります。その AWS リージョンではスタンドアロンクラスターとして残ります。
 - [Disassociated] - セカンダリクラスターは Global Datastore から削除され、その AWS リージョンでスタンドアロンクラスターになりました。
- グローバルデータストアレプリカの遅延 — グローバルデータストアのセカンダリ AWS リージョンごとに 1 つの値を表示します。これは、セカンダリリージョンのプライマリノードとプライマリリージョンのプライマリノード間の遅延です。クラスターモードが有効な Redis の場合、遅延はシャード間の最大遅延 (秒単位) を示します。

Global Datastore へのリージョンの追加

既存のグローバルデータストアには、最大 1 つの AWS リージョンを追加できます。このシナリオでは、プライマリクラスターから自動更新と非同期更新を受け取る読み取り専用クラスターを別の AWS リージョンに作成します。

AWS リージョンをグローバルデータストアに追加するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択し、既存のグローバルデータストア を選択します。
3. リージョンクラスターの追加 を選択し、セカンダリクラスターが存在する AWS リージョンを選択します。
4. クラスター情報 で、名前 に値を入力し、オプションでクラスターの説明 に値を入力します。
5. 次のオプションをそのまま使用します。これらは、プライマリクラスターの設定に合わせて事前に設定されていて、変更することはできません。

- エンジンバージョン
- インスタンスタイプ
- ノードの種類
- シャード数
- パラメータグループ

Note

ElastiCache は、指定されたパラメータグループの値から新しいパラメータグループを自動生成し、新しいパラメータグループをクラスターに適用します。この新しいパラメータグループを使用して、Global Datastore のパラメータを変更します。自動生成された各パラメータグループは、1 つの唯一のクラスターにのみ関連付けられます。したがって、1 つの唯一の Global Datastore にのみ関連付けられます。

- 保管中の暗号化

Note

カスタマーマネージド AWS KMS キーを選択し、そのキーを選択することで、別の暗号化キーを指定できます。

- 転送中の暗号化
 - Redis AUTH
6. (オプション) セカンダリクラスターの残りの設定を更新します。プライマリクラスターと同じ値が事前に入力されますが、そのクラスターの特定の要件を満たすように設定を更新できます。
- [ポート]
 - レプリケーション数
 - サブネットグループ
 - 優先アベイラビリティゾーン
 - セキュリティグループ
 - カスタマーマネージド AWS KMS キー)
 - Redis AUTH トークン
 - 自動バックアップの有効化
 - バックアップの保存期間
 - バックアップウィンドウ
 - メンテナンスウィンドウ
 - SNS 通知のトピック
7. [追加] を選択します。

Global Datastore の変更

リージョンクラスターのプロパティを変更できます。Global Datastore で実行できる変更オペレーションは 1 つだけです。ただし、セカンダリクラスターをプライマリに昇格させることは例外です。詳細については、「[セカンダリクラスターのプライマリへの昇格](#)」を参照してください。

Global Datastore を変更するには

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

2. ナビゲーションペインでグローバルデータストアを選択し、グローバルデータストア名でグローバルデータストアを選択します。
3. [Modify] を選択し、次のオプションの中から選択します。
 - [Modify description] – Global Datastore の説明を更新します
 - [Modify engine version] – Redis エンジンバージョン 5.0.6 以降のみ使用できます。
 - [Modify node type] – リージョンクラスターを垂直方向 (スケールアップとスケールダウン) と水平方向 (スケールインとスケールアウト) の両方にスケールします。オプションには、R5 および M5 ノードファミリーがあります。ノードタイプの詳細については、「[サポートされているノードの種類](#)」を参照してください。
 - [自動フェイルオーバーの変更] – 自動フェイルオーバーを有効または無効にします。リージョンクラスターでフェイルオーバーとプライマリノードを有効にすると、はリージョンレプリカの 1 つに ElastiCache フェイルオーバーします。詳細については、「[Auto Failover](#)」を参照してください。

クラスターモードが有効になっている Redis クラスターの場合:

- [Add shards] – 追加するシャードの数を入力し、オプションで 1 つ以上のアベイラビリティーゾーンを指定します。
- シャードの削除 – 各 AWS リージョンで削除するシャードを選択します。
- [Rebalance shards] – スロット配分を再分散して、クラスター内の既存のシャード間で均一な分散を確保します。

グローバルデータストアのパラメータを変更するには、グローバルデータストアの任意のメンバークラスターのパラメータグループを変更します。は、この変更をそのグローバルデータストア内のすべてのクラスターに自動的に ElastiCache 適用します。そのクラスターのパラメータグループを変更するには、Redis コンソールまたは [ModifyCacheCluster](#) API オペレーションを使用します。詳細については、「[パラメータグループを変更する](#)」を参照してください。Global Datastore に含まれているクラスターのパラメータグループを変更すると、その Global Datastore 内のすべてのクラスターに適用されます。

パラメータグループ全体または特定のパラメータをリセットするには、[ResetCacheParameterGroup](#) API オペレーションを使用します。

セカンダリクラスターのプライマリへの昇格

プライマリクラスターまたは AWS リージョンが使用できなくなった場合、またはパフォーマンスの問題が発生した場合は、セカンダリクラスターをプライマリに昇格させることができます。昇格は、他の変更が進行中であっても、いつでも許可されます。複数のプロモーションを並行して発行することもできます。Global Datastore は、最終的に 1 つのプライマリに解決されます。複数のセカンダリクラスターを同時に昇格させた場合、ElastiCache For Redis は最終的にプライマリクラスターに解決されるクラスターを保証しません。

セカンダリクラスターをプライマリクラスターに昇格するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択します。
3. Global Datastore 名を選択して詳細を表示します。
4. [Secondary] クラスターを選択します。
5. [Promote to primary] を選択します。

続いて、次の警告が表示され、決定を確認するように求められます: Promoting a region to primary will make the cluster in this region as read/writable. Are you sure you want to promote the *secondary* cluster to primary?.

The current primary cluster in *primary region* will become secondary and will stop accepting writes after this operation completes. Please ensure you update your application stack to direct traffic to the new primary region.

6. 昇格を続行する場合は [確認] を、続行しない場合は [キャンセル] を選択します。

確認を選択した場合、Global Datastore は [Modifying] 状態に変わり、昇格が完了するまで使用できなくなります。

Global Datastore からのリージョンの削除

次の手順を使用して、グローバルデータストアから AWS リージョンを削除できます。

グローバルデータストアから AWS リージョンを削除するには

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択します。
3. [Global Datastore] を選択します。
4. 削除する [Region] を選択します。
5. [Remove region] を選択します。

Note

このオプションは、セカンダリクラスターでのみ使用できます。

続いて、次の警告が表示され、決定を確認するように求められます: Removing the region will remove your only available cross region replica for the primary cluster. Your primary cluster will no longer be set up for disaster recovery and improved read latency in remote region. Are you sure you want to remove the selected region from the global datastore?。

6. 昇格を続行する場合は [確認] を、続行しない場合は [キャンセル] を選択します。

確認を選択すると、AWS リージョンが削除され、セカンダリクラスターはレプリケーションの更新を受信しなくなります。

Global Datastore の削除

Global Datastore を削除するには、まずすべてのセカンダリクラスターを削除します。詳細については、「[Global Datastore からのリージョンの削除](#)」を参照してください。これにより、Global Datastore のステータスは [primary-only] になります。

Global Datastore を削除するには

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、グローバルデータストア を選択します。
3. [Global Datastore Name] で、削除する Global Datastore を選択し、[削除] を選択します。

続いて、次の警告が表示され、決定を確認するように求められます: Are you sure you want to delete this Global Datastore?。

4. [削除] を選択します。

Global Datastore が [Deleting] ステータスに変わります。

Global Datastore の使用 (CLI)

AWS Command Line Interface (AWS CLI) を使用すると、複数の AWS のサービスをコマンドラインから制御したり、スクリプトで自動化したりできます。AWS CLI は、アドホック (ワンタイム) オペレーションに使用できます。

AWS CLI のダウンロードと設定

AWS CLI は、Windows、macOS または Linux 上で作動します。これをダウンロードして設定するには、次の手順に従います。

CLI をダウンロード、インストール、設定するには

1. [\[AWS コマンドラインインターフェイス\]](#) のウェブページで AWS CLI をダウンロードします。
2. [\[AWS Command Line Interface ユーザーガイド\]](#) の AWS CLI のインストールおよび AWS の CLI の設定の手順に従います。

Global Datastore での AWS CLI の使用

Global Datastore を操作するには、次の CLI オペレーションを使用します。

- [create-global-replication-group](#)

```
aws elasticache create-global-replication-group \  
  --global-replication-group-id-suffix my global datastore \  
  --primary-replication-group-id sample-repl-group \  
  --global-replication-group-description an optional description of the global  
datastore
```

Amazon ElastiCache は、Global Datastore ID の作成時にプレフィックスを自動的に適用します。各 AWS リージョンには独自のプレフィックスがあります。例えば、米国西部 (北カルフォルニア) リージョンで作成された Global Datastore ID は、指定したサフィックス名と共に「virxk」で始ま

ります。サフィックスは、自動生成されたプレフィックスと組み合わせられて、複数のリージョンにまたがるグローバルデータストア名の一意性を保証します。

以下のテーブルは、各 AWS リージョンとその Global Datastore ID プレフィックスを一覧表示します。

リージョン名/リージョン	プレフィックス
米国東部 (オハイオ) リージョン us-east-2	fpkhr
米国東部(バージニア州北部) リージョン us-east-1	ldgnf
US West (N. California) Region us-west-1	virxk
米国西部 (オレゴン) リージョン us-west-2	sgau
カナダ (中部) リージョン ca-central-1	bxodz
アジアパシフィック (ムンバイ) リージョン ap-south-1	erpgt
アジアパシフィック (東京) リージョン ap-northeast-1	qusw
アジアパシフィック (ソウル) リージョン ap-northeast-2	lfqnh

リージョン名/リージョン	プレフィックス
アジアパシフィック (大阪) リージョン ap-northeast-3	nlapn
アジアパシフィック (シンガポール) リージョン ap-southeast-1	vlqxn
アジアパシフィック (シドニー) リージョン ap-southeast-2	vbgxd
欧州 (フランクフルト) リージョン eu-central-1	iudkw
欧州 (アイルランド) リージョン eu-west-1	gxeiz
欧州 (ロンドン) リージョン eu-west-2	okuqm
EU (パリ) リージョン eu-west-3	fgjhi
南米 (サンパウロ) リージョン sa-east-1	juxlw
中国 (北京) リージョン cn-north-1	emvgo
中国 (寧夏) リージョン cn-northwest-1	ckbem

リージョン名/リージョン	プレフィックス
アジアパシフィック (香港) リージョン ap-east-1	knjmp
AWS GovCloud (米国西部) us-gov-west-1	sgwui

- [\[create-replication-group\]](#) – このオペレーションを使用して、Global Datastore の名前を `--global-replication-group-id` パラメータに指定することで、Global Datastore のセカンダリクラスターを作成します。

```
aws elasticache create-replication-group \
  --replication-group-id secondary replication group name \
  --replication-group-description "Replication group description" \
  --global-replication-group-id global datastore name
```

このオペレーションを呼び出して `--global-replication-group-id` 値を受け渡した場合、ElastiCache for Redis は、次のパラメータについて、グローバルレプリケーショングループのプライマリレプリケーショングループから値を推測します。これらのパラメータには値を渡さないでください。

"PrimaryClusterId",

"AutomaticFailoverEnabled",

"NumNodeGroups",

"CacheParameterGroupName",

"CacheNodeType",

"Engine",

"EngineVersion",

"CacheSecurityGroupNames",

"EnableTransitEncryption",
"AtRestEncryptionEnabled",
"SnapshotArns",
"SnapshotName"

- [describe-global-replication-groups](#)

```
aws elasticache describe-global-replication-groups \  
  --global-replication-group-id my global datastore \  
  --show-member-info an optional parameter that returns a list of the primary and  
  secondary clusters that make up the global datastore
```

- [modify-global-replication-group](#)

```
aws elasticache modify-global-replication-group \  
  --global-replication-group-id my global datastore \  
  --automatic-failover-enabled \  
  --cache-node-type node type \  
  --cache-parameter-group-name parameter group name \  
  --engine-version engine version \  
  --apply-immediately \  
  --global-replication-group-description description
```

- [delete-global-replication-group](#)

```
aws elasticache delete-global-replication-group \  
  --global-replication-group-id my global datastore \  
  --retain-primary-replication-group defaults to true
```

- [disassociate-global-replication-group](#)

```
aws elasticache disassociate-global-replication-group \  
  --global-replication-group-id my global datastore \  
  --replication-group-id my secondary cluster \  
  --replication-group-region the AWS Region in which the secondary cluster resides
```

- [failover-global-replication-group](#)


```
aws elasticache failover-replication-group \  
  --global-replication-group-id my global datastore \  
  --primary-region The AWS Region of the primary cluster \  
  --primary-replication-group-id The name of the global datastore, including the  
  suffix.
```

- [increase-node-groups-in-global-replication-group](#)

```
aws elasticache increase-node-groups-in-global-replication-group \  
  --apply-immediately yes \  
  --global-replication-group-id global-replication-group-name \  
  --node-group-count 3
```

- [decrease-node-groups-in-global-replication-group](#)

```
aws elasticache decrease-node-groups-in-global-replication-group \  
  --apply-immediately yes \  
  --global-replication-group-id global-replication-group-name \  
  --node-group-count 3
```

- [rebalance-shards-in-global-replication-group](#)

```
aws elasticache rebalance-shards-in-global-replication-group \  
  --apply-immediately yes \  
  --global-replication-group-id global-replication-group-name
```

ヘルプを使用すると、ElastiCache for Redis で使用可能なすべてのコマンドを列挙できます。

```
aws elasticache help
```

また、ヘルプを使用して、特定コマンドを記述したり、その用法の詳細を確認したりすることもできます。

```
aws elasticache create-global-replication-group help
```

レプリケーショングループを使用する高可用性

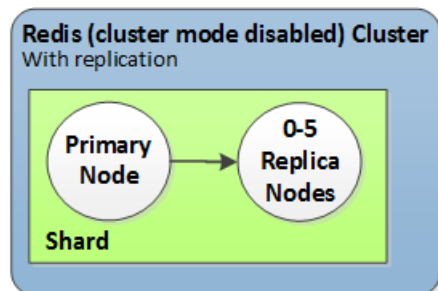
単一ノードの Amazon ElastiCache Redis クラスターは、限られたデータ保護サービス (AOF) を持つインメモリエンティティです。クラスターが何らかの理由で停止すると、クラスターのすべてのデータが失われます。ただし Redis エンジンを実行している場合は、2~6 個のノードをクラスターとしてグループ化できます。この場合、1~5 個の読み取り専用ノードは、グループに 1 個含まれる読み書きプライマリノードのレプリカデータです。このシナリオでは、1 個のノードが何らかの理由で停止した場合でも 1 個以上の他のノードにレプリケートされているので、すべてのデータが失われることはありません。レプリケーションのレイテンシーが原因でプライマリの読み取り/書き込みノードが失敗した場合、一部のデータが失われる可能性があります。

次の図に示されているように、レプリケーション構造体は、Redis クラスター内に含まれている (API/CLI の [ノードグループ] という名前の) シャードに含まれています。Redis (クラスターモードが無効) クラスターには、常に 1 つのシャードがあります。Redis (クラスターモードが有効) クラスターには、クラスターのデータがシャード間で分割された状態で最大 500 個のシャードを持つことができます。シャードの数が多くレプリカ数が少ないクラスターを作成できます。クラスターあたり最大 90 ノードです。このクラスター設定は、シャード 90 個およびレプリカ 0 個からシャード 15 個およびレプリカ 5 個 (許容されるレプリカの最大数) までです。

Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。



Redis (クラスターモードが無効) クラスターに、1つのシャードと0から5のレプリカノードがあるマルチAZが有効になっているクラスターにレプリカがある場合、プライマリノードで障害が発生すると、プライマリはリードレプリカにフェイルオーバーします。データがレプリカノードに非同期で更新されるため、レプリカノードの更新のレイテンシーにより多少のデータが失われる場合があります。詳細については、「[Redis 実行時の障害の軽減](#)」を参照してください。

トピック

- [Redis レプリケーションについて](#)
- [レプリケーション: Redis \(クラスターモードが無効\) 対 Redis \(クラスターモードが有効\)](#)
- [マルチAZによるElastiCache for Redisのダウンタイムの最小化](#)
- [同期とバックアップの実装方法](#)
- [Redis レプリケーショングループの作成](#)
- [レプリケーショングループの詳細の表示](#)
- [レプリケーショングループのエンドポイントの検索](#)
- [レプリケーショングループの変更](#)
- [レプリケーショングループの削除](#)
- [レプリカの数の変更](#)
- [Redis \(クラスターモード無効\) レプリケーショングループのリードレプリカのプライマリへの昇格](#)

Redis レプリケーションについて

Redis では、次の 2 つの方法でレプリケーションが実装されます。

- クラスターのすべてのデータを各ノードに含む単一シャード—Redis (クラスターモードが無効)
- 最大 500 個のシャード間でデータを分割する Redis (クラスターモードが有効)

レプリケーショングループ内の各シャードには、単一の読み取り/書き込みプライマリノードと、最大 5 個の読み取り専用レプリカノードがあります。シャードの数が多くレプリカの数が少ないクラスターを作成できます。クラスターあたり最大 90 ノードです。このクラスター設定は、シャード 90 個およびレプリカ 0 個からシャード 15 個およびレプリカ 5 個 (許容されるレプリカの最大数) までです。

Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

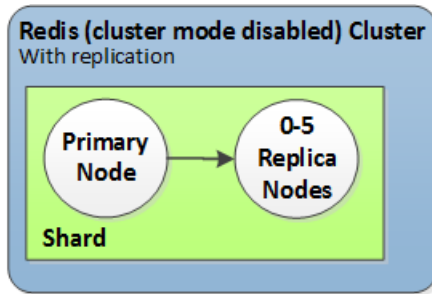
トピック

- [Redis \(クラスターモードが無効\)](#)
- [Redis \(クラスターモードが有効\)](#)

Redis (クラスターモードが無効)

Redis (クラスターモードが無効) クラスターには 1 つのシャードがあり、その内部は Redis ノードの集合で、1 個の読み取り/書き込みプライマリノードと、最大 5 個の読み取り専用のセカンダリレプリカノードで構成されます。各リードレプリカは、クラスターのプライマリノードにあるデータのコピーを保持します。非同期レプリケーション機能は、リードレプリカとプライマリの同期を維持するのに使用されます。アプリケーションは、クラスター内のどのノードからでも読み取ることが

できます。アプリケーションは、そのプライマリノードにのみ書き込むことができます。リードレプリカは読み取りスループットを向上させ、ノードの障害発生時のデータ損失に対する保護を強化します。



1つのシャードと複数のレプリカノードのある Redis (クラスターモードが無効) クラスター

レプリカノードで Redis (クラスターモードが無効) クラスターを使用すると、の ElastiCache Redis ソリューションをスケールして、読み取りを多用するアプリケーションを処理するか、同じクラスターから同時に読み取りを行う多数のクライアントをサポートできます。

Redis (クラスターモードが無効) クラスター内のすべてのノードは、同じリージョンに存在する必要があります。

クラスターにリードレプリカを追加すると、プライマリのすべてのデータが新しいノードにコピーされます。その時以降、データがプライマリに書き込まれるときには常に、変更が非同期的にすべてのリードレプリカに反映されます。

耐障害性を向上させて書き込みのダウンタイムを減少させるには、レプリカを持つ Redis (クラスターモードが無効) クラスターで自動フェイルオーバーを備えたマルチ AZ を有効にします。詳細については、「[マルチ AZ による ElastiCache for Redis のダウンタイムの最小化](#)」を参照してください。

Redis (クラスターモードが無効) クラスター内のノードのロールを変更し、プライマリといずれかのレプリカのロールを交換できます。この作業は、パフォーマンスチューニングの理由で実行することがあります。たとえば、書き込みアクティビティが多いウェブアプリケーションでは、ネットワークレイテンシーが最も低いノードを選択することができます。詳細については、「[Redis \(クラスターモード無効\) レプリケーショングループのリードレプリカのプライマリへの昇格](#)」を参照してください。

Redis (クラスターモードが有効)

Redis (クラスターモードが有効) クラスターは、1~500 個のシャード (API/CLI: ノードグループ) で構成されます。各シャードには、読み取り/書き込みプライマリノードと最大 5 個のリードレプリカ

ノードが含まれます。この構成は、シャード 90 個およびレプリカ 0 個からシャード 15 個およびレプリカ 5 個 (許容されるレプリカの最大数) までです。

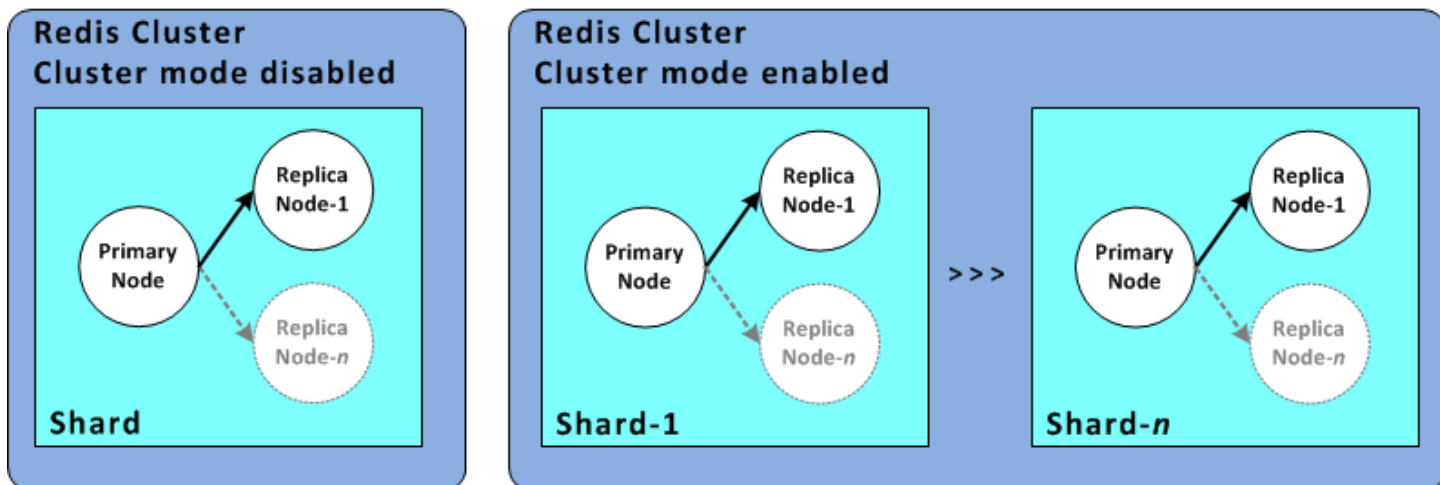
Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

シャード内の各リードレプリカは、シャードのプライマリからのデータのコピーを維持します。非同期レプリケーション機能は、リードレプリカとプライマリの同期を維持するのに使用されます。アプリケーションは、クラスター内のどのノードからでも読み取ることができます。アプリケーションは、そのプライマリノードにのみ書き込むことができます。リードレプリカは、読み取り拡張性およびデータ損失に対する保護を強化します。データは、Redis (クラスターモードが有効) クラスター内のシャード間で分割されます。

アプリケーションは、Redis (クラスターモードが有効) クラスターの [設定エンドポイント] を使用して、クラスター内のノードと接続します。詳細については、「[接続エンドポイントの検索](#)」を参照してください。



複数のシャードとレプリカノードのある Redis (クラスターモードが有効) クラスター

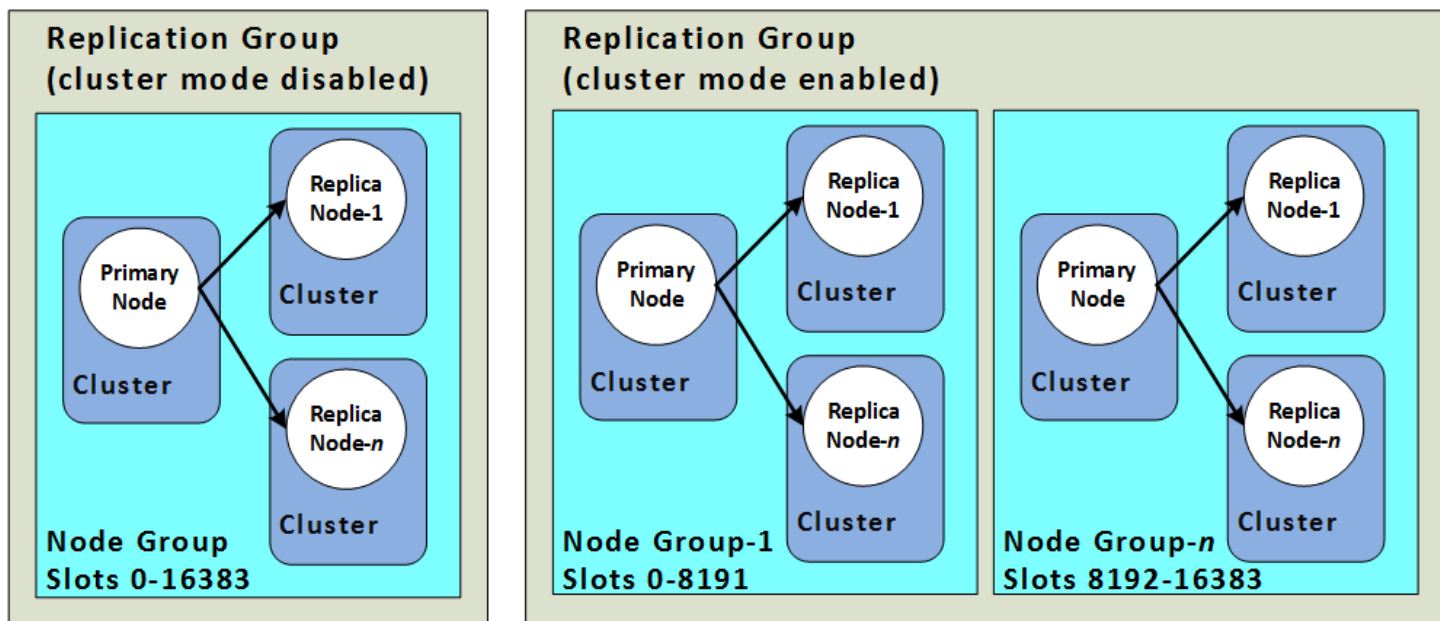
Redis (クラスターモードが有効) クラスター内のすべてのノードは、同じリージョンに存在する必要があります。耐障害性を向上させるために、そのリージョン内の複数のアベイラビリティゾーンにプライマリとリードレプリカの両方をプロビジョニングできます。

現在、Redis (クラスターモードが有効) には、いくつかの制限があります。

- いずれのレプリカノードも手動でプライマリに昇格することはできません。

レプリケーション: Redis (クラスターモードが無効) 対 Redis (クラスターモードが有効)

Redis バージョン 3.2 以降では、2 つの異なるタイプの Redis クラスター (API/CLI: レプリケーショングループ) のいずれかを作成できます。Redis (クラスターモードが無効) クラスターは常に 1 つのシャード (API/CLI: ノードグループ) と最大 5 個のリードレプリカノードで構成されます。それぞれに 1~5 個のリードレプリカノードのある最大 500 個のシャードを持つ Redis (クラスターモードが有効) クラスター。



Redis (クラスターモードが無効) と Redis (クラスターモードが有効) クラスター

次の表に、Redis (クラスターモードが無効) クラスターと Redis (クラスターモードが有効) クラスターの重要な違いをまとめます。

Redis (クラスターモードが無効) と Redis (クラスターモードが有効) クラスターの比較

機能	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
変更可能	はい。レプリカノードの追加と削除、およびノードタイプのスケールアップをサポートします。	制限あり。詳細については、「 エンジンバージョンとアップグレード 」および「 Redis (クラスターモードが有効) でのクラスターのスケールアップ 」を参照してください。
データのパーティション化	いいえ	はい
シャード	1	1~500
リードレプリカ	0~5 ⚠ Important レプリカがない場合、ノードに障害が発生すると、すべてのデータが損失します。	シャードあたり 0~5。 ⚠ Important レプリカがなく、ノードに障害が発生すると、そのシャードのすべてのデータが失われます。
マルチ AZ	はい、少なくとも 1 つのレプリカ。 オプション。デフォルトでオン。	はい オプション。デフォルトでオン。
スナップショット(バックアップ)	はい、1 つの .rdb ファイルを作成。	はい、シャードごとに独自の .rdb ファイルを作成。
復元	はい。Redis (クラスターモードが無効) クラスターから 1 つの .rdb ファイルを使用します。	はい。Redis (クラスターモードが無効) または Redis (クラスターモードが有効) クラス

機能	Redis (クラスターモードが無効)	Redis (クラスターモードが有効) ターから.rdb ファイルを使用します。
サポート	Redis のすべてのバージョン	Redis 3.2 以降
エンジンがアップグレード可能	はい。ただし、いくつかの制限があります。詳細については、「 エンジンバージョンとアップグレード 」を参照してください。	はい。ただし、いくつかの制限があります。詳細については、「 エンジンバージョンとアップグレード 」を参照してください。
暗号化	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。
HIPAA 適格	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。
PCI DSS 準拠	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。	バージョン 3.2.6 (EOL の予定、「 Redis バージョンのサポート終了スケジュール 」を参照) および 4.0.10 以降。
オンラインリシャードニング	該当なし	バージョン 3.2.10 (EOL の予定。「 Redis バージョンのサポート終了スケジュール 」を参照) 以降。

どちらを使用すればよいですか？

Redis (クラスターモードが無効) または Redis (クラスターモードが有効) のいずれかを選択するときは、次の要素を考慮してください。

- [スケーリングとパーティション化] – ビジネスには変化が必要です。ピーク需要に対してプロビジョニングするか、需要の変化に応じてスケールする必要があります。Redis (クラスターモードが無効) は、スケーリングをサポートしています。レプリカノードを追加または削除して読み取り容量をスケールするか、より大きいノードタイプにスケールアップして容量をスケールできます。両方のオペレーションには時間がかかります。詳細については、[レプリカノードを含む Redis \(クラスターモードが無効\) クラスターのスケーリング](#) を参照してください。

Redis (クラスターモードが有効) では、最大 500 個のノードグループ間でのデータの分割がサポートされています。ビジネスニーズの変化に合わせて、シャードの数を動的に変更することができます。パーティション化の 1 つの利点は、より多くのエンドポイントに負荷を分散し、ピーク需要時のアクセスのボトルネックを減らすことです。また、データを複数のサーバーに分散させることができるため、より大規模なデータセットに対応できます。パーティションのスケーリングについては、「」を参照してください[Redis \(クラスターモードが有効\) でのクラスターのスケーリング](#)。

- [ノードサイズとノード数] – Redis (クラスターモードが無効) クラスターのシャードは 1 つだけであるため、ノードタイプはクラスターのすべてのデータと必要なオーバーヘッドに対応できるだけの大きさである必要があります。一方、Redis (クラスターモードが有効) クラスターでは複数のシャード間でデータを分割できるため、ノードタイプはより小さくできます。ただし、全体ではノード数が増えます。詳細については、「[ノードサイズの選択](#)」を参照してください。
- [読み取りと書き込み] – クラスターの主な負荷がデータを読み取るアプリケーションである場合、リードレプリカを追加および削除することで、Redis (クラスターモードが無効) クラスターをスケールできます。ただし、リードレプリカの最大数は 5 です。クラスターの書き込み負荷が高い場合は、複数のシャードを持つ Redis (クラスターモードが有効) クラスターの追加の書き込みエンドポイントが役立ちます。

どちらのクラスターを実装する場合でも、現在および将来のニーズに合ったノードタイプを選択してください。

マルチ AZ による ElastiCache for Redis のダウンタイムの最小化

ElastiCache for Redis がプライマリノードを置き換える必要があるインスタンスは多数あります。これには、特定のタイプの計画されたメンテナンスや、プライマリノードまたはアベイラビリティーゾーンの障害が発生する可能性が低いことが含まれます。

この置き換えにより、クラスターのダウンタイムが発生しますが、マルチ AZ が有効になっている場合、ダウンタイムは最小限に抑えられます。プライマリノードのロールは、いずれかのリードレプリカに自動的にフェイルオーバーされます。はこれを透過的に ElastiCache 処理するため、新しいプライマリノードを作成してプロビジョニングする必要はありません。このフェイルオーバーとレプリカの昇格により、昇格が完了したらすぐに新しいプライマリへの書き込みを再開できます。

ElastiCache は、昇格されたレプリカのドメインネームサービス (DNS) 名も伝播します。これを行うのは、アプリケーションがプライマリエンドポイントに書き込みを行う場合、アプリケーションでエンドポイントの変更が必要なくなるためです。個別のエンドポイントから読み取りを行う場合は、プライマリに昇格されたレプリカの読み取りエンドポイントを新しいレプリカのエンドポイントに変更してください。

メンテナンス更新やセルフサービス更新に伴って開始された計画的なノード置換の場合:

- ElastiCache for Redis クラスターの場合、クラスターが受信書き込みリクエストを処理する間に、計画されたノード交換が完了します。
- Redis クラスターモードが無効で、マルチ AZ が有効になっているクラスターが 5.0.6 以降のエンジンで実行されている場合、クラスターが着信した書き込みリクエストを処理している間に、計画されたノード置換が完了します。
- Redis クラスターモードが無効で、マルチ AZ が有効になっているクラスターで 4.0.10 以前のエンジンで実行されている場合、DNS の更新に伴って短い書き込みの中断が発生することがあります。この中断は数秒続く場合があります。このプロセスは、新しいプライマリを再作成してプロビジョニングする (マルチ AZ を有効にしない場合に発生すること) よりもはるかに高速です。

マルチ AZ は、ElastiCache マネジメントコンソール、AWS CLI、または ElastiCache API を使用して有効にできます。

Redis クラスター (API および CLI、レプリケーショングループ) で ElastiCache マルチ AZ を有効にすると、耐障害性が向上します。これは特に、クラスターの読み取り/書き込みプライマリクラスターノードが到達できなくなった場合、または何らかの理由で障害が発生した場合に当てはまります。マルチ AZ は、各シャードに複数のノードがある Redis クラスターでのみサポートされます。

トピック

- [マルチ AZ の有効化](#)
- [障害シナリオとマルチ AZ のレスポンス](#)
- [自動フェイルオーバーのテスト](#)
- [Redis のマルチ AZ の制限事項](#)

マルチ AZ の有効化

ElastiCache コンソール、または API を使用してクラスター (API または CLI、レプリケーショングループ) を作成または ElastiCache 変更するときに AWS CLI、マルチ AZ を有効にできます。

マルチ AZ は、使用可能なリードレプリカが少なくとも 1 つある Redis (クラスターモードが無効) クラスターでのみ有効にすることができます。リードレプリカのないクラスターでは、高可用性や耐障害性は提供されません。レプリケーションが有効なクラスターの作成については、「[Redis レプリケーショングループの作成](#)」を参照してください。レプリケーションが有効なクラスターへのリードレプリカの追加については、「[Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの追加](#)」を参照してください。

トピック

- [マルチ AZ の有効化 \(コンソール\)](#)
- [マルチ AZ の有効化 \(AWS CLI\)](#)
- [マルチ AZ の有効化 \(ElastiCache API\)](#)

マルチ AZ の有効化 (コンソール)

新しい Redis クラスターを作成するとき、またはレプリケーションで既存の Redis クラスターを変更することで、ElastiCache コンソールを使用してマルチ AZ を有効にできます。

マルチ AZ は、Redis (クラスターモードが有効) クラスターでデフォルトで有効になります。

Important

ElastiCache は、クラスターにすべてのシャードのプライマリとは異なるアベイラビリティゾーンに少なくとも 1 つのレプリカが含まれている場合にのみ、マルチ AZ を自動的に有効にします。

ElastiCache コンソールを使用してクラスターを作成するときにマルチ AZ を有効にする

このプロセスの詳細については、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」を参照してください。必ず 1 つ以上のレプリカを用意して、マルチ AZ を有効にしてください。

既存のクラスターでのマルチ AZ の有効化 (コンソール)

このプロセスの詳細については、「[の使用 AWS Management Console](#)」でクラスターの変更に関する説明を参照してください。

マルチ AZ の有効化 (AWS CLI)

次のコード例では、を使用して AWS CLI レプリケーショングループのマルチ AZ を有効にします redis12。

⚠ Important

レプリケーショングループ redis12 が既に存在しており、少なくとも 1 個の利用可能なリードレプリカが必要となります。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id redis12 \  
  --automatic-failover-enabled \  
  --multi-az-enabled \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id redis12 ^  
  --automatic-failover-enabled ^  
  --multi-az-enabled ^  
  --apply-immediately
```

このコマンドの JSON 出力は次のようになります。

```
{  
  "ReplicationGroup": {
```

```
"Status": "modifying",
"Description": "One shard, two nodes",
"NodeGroups": [
  {
    "Status": "modifying",
    "NodeGroupMembers": [
      {
        "CurrentRole": "primary",
        "PreferredAvailabilityZone": "us-west-2b",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address":
"redis12-001.v5r9dc.0001.usw2.cache.amazonaws.com"
        },
        "CacheClusterId": "redis12-001"
      },
      {
        "CurrentRole": "replica",
        "PreferredAvailabilityZone": "us-west-2a",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address":
"redis12-002.v5r9dc.0001.usw2.cache.amazonaws.com"
        },
        "CacheClusterId": "redis12-002"
      }
    ],
    "NodeGroupId": "0001",
    "PrimaryEndpoint": {
      "Port": 6379,
      "Address": "redis12.v5r9dc.ng.0001.usw2.cache.amazonaws.com"
    }
  }
],
"ReplicationGroupId": "redis12",
"SnapshotRetentionLimit": 1,
"AutomaticFailover": "enabling",
"MultiAZ": "enabled",
"SnapshotWindow": "07:00-08:00",
"SnapshottingClusterId": "redis12-002",
"MemberClusters": [
  "redis12-001",
```

```
        "redis12-002"  
    ],  
    "PendingModifiedValues": {}  
}  
}
```

詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。

- [create-cache-cluster](#)
- [create-replication-group](#)
- AWS CLI コマンドリファレンスの [modify-replication-group](#)。

マルチ AZ の有効化 (ElastiCache API)

次のコード例では、ElastiCache API を使用してレプリケーショングループのマルチ AZ を有効にします redis12。

Note

この例を使用するには、レプリケーショングループ redis12 が既に存在していて、少なくとも 1 個の利用可能なリードレプリカがある必要があります。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyReplicationGroup  
&ApplyImmediately=true  
&AutoFailover=true  
&MultiAZEnabled=true  
&ReplicationGroupId=redis12  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20140401T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、ElastiCache API リファレンスの以下のトピックを参照してください。

- [CreateCacheクラスター](#)
- [CreateReplicationグループ](#)

- [ModifyReplicationグループ](#)

障害シナリオとマルチ AZ のレスポンス

マルチ AZ の導入前に、は、障害が発生したノードを再作成して再プロビジョニングすることで、クラスタの障害が発生したノード ElastiCache を検出して置き換えました。マルチ AZ を有効にすると、失敗したプライマリノードはレプリケーションの遅延が最も小さいレプリカにフェイルオーバーされます。選択されたレプリカは自動的にプライマリに昇格されます。このプロセスは、新しいプライマリノードを作成して再プロビジョニングするよりも大幅に高速です。通常は数秒で、クラスタへの書き込みが再び可能になります。

マルチ AZ が有効になっている場合は、プライマリノードの状態 ElastiCache を継続的にモニタリングします。プライマリノードが失敗すると、失敗のタイプに応じて次のいずれかのアクションが実行されます。

トピック

- [プライマリノードのみが失敗した場合の障害シナリオ](#)
- [プライマリノードと複数のリードレプリカが失敗した場合の障害シナリオ](#)
- [クラスタ全体が失敗した場合の障害シナリオ](#)

プライマリノードのみが失敗した場合の障害シナリオ

プライマリノードのみが失敗した場合、レプリケーションの遅延が最も小さいリードレプリカがプライマリに昇格されます。次に、失敗したプライマリと同じアベイラビリティゾーンに置換リードレプリカが作成されてプロビジョニングされます。

プライマリノードのみが失敗した場合、ElastiCache マルチ AZ は以下を実行します。

1. 失敗したプライマリノードがオフラインになります。
2. レプリケーションの遅延が最短のリードレプリカがプライマリに昇格されます。

書き込みは、昇格プロセスが完了するとすぐに (通常は数秒) 再開できます。アプリケーションがプライマリエンドポイントに書き込む場合、書き込みまたは読み取りのエンドポイントを変更する必要はありません。ElastiCache は昇格されたレプリカの DNS 名を伝播します。

3. 置き換えられたリードレプリカが起動し、プロビジョニングされます。

ノードのディストリビューションが維持されるように、障害が発生したプライマリノードがあったアベイラビリティゾーンで置き換えリードレプリカが起動されます。

4. レプリカが新しいプライマリノードと同期されます。

新しいレプリカが使用可能になった後は、次の影響に注意してください。

- [プライマリエンドポイント] – 新しいプライマリノードの DNS 名がプライマリエンドポイントに伝達されるため、アプリケーションに変更は加えません。
- [読み取りエンドポイント] – 読み取りエンドポイントは、新しいレプリカノードを指すように自動的に更新されます。

クラスターのエンドポイントの検索については、以下のトピックを参照してください。

- [Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)
- [レプリケーショングループのエンドポイントの検索 \(AWS CLI\)](#)
- [レプリケーショングループのエンドポイントの検索 \(ElastiCache API\)](#)

プライマリノードと複数のリードレプリカが失敗した場合の障害シナリオ

プライマリおよび少なくとも 1 つのリードレプリカで障害が発生した場合、利用可能でレプリケーションの遅延が最も少ないレプリカが、プライマリクラスターに昇格されます。また、障害が発生したノードおよびプライマリに昇格されたレプリカと同じアベイラビリティゾーンで、新しいリードレプリカが作成およびプロビジョニングされます。

プライマリノードと一部のリードレプリカが失敗すると、ElastiCache マルチ AZ は以下を実行します。

1. 障害が発生したプライマリノードとリードレプリカがオフラインになります。
2. レプリケーションの遅延が最短の使用可能なレプリカがプライマリノードに昇格されます。

書き込みは、昇格プロセスが完了するとすぐに (通常は数秒) 再開できます。アプリケーションがプライマリエンドポイントに書き込む場合、writes. ElastiCache propagates the DNS name of the promoted replica のエンドポイントを変更する必要はありません。

3. 複数の置き換えレプリカを作成してプロビジョニングします。

ノードのディストリビューションが維持されるように、障害が発生したノードのアベイラビリティゾーンで置き換えレプリカが作成されます。

4. すべてのクラスターが新しいプライマリノードと同期されます。

新しいノードが使用可能になったら、アプリケーションに以下の変更を行います。

- [プライマリエンドポイント] – アプリケーションは変更しないでください。新しいプライマリノードの DNS 名がプライマリエンドポイントに伝達されます。
- [読み取りエンドポイント] – 読み取りエンドポイントは、新しいレプリカノードを指すように自動的に更新されます。

レプリケーショングループのエンドポイントの検索については、次のトピックを参照してください:

- [Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)
- [レプリケーショングループのエンドポイントの検索 \(AWS CLI\)](#)
- [レプリケーショングループのエンドポイントの検索 \(ElastiCache API\)](#)

クラスター全体が失敗した場合の障害シナリオ

すべてに障害が発生した場合、すべてのノードは、元のノードと同じアベイラビリティーゾーンで再作成され、プロビジョニングされます。

このシナリオでは、クラスター内のすべてのデータがクラスター内のすべてのノードの障害のために失われます。これはまれにしか発生しません。

クラスター全体が失敗すると、ElastiCache マルチ AZ は以下を実行します。

1. 障害が発生したプライマリノードとリードレプリカがオフラインになります。
2. 置き換えプライマリノードが作成され、プロビジョニングされます。
3. 複数の置き換えレプリカを作成してプロビジョニングします。

ノードのディストリビューションが維持されるように、障害が発生したノードのアベイラビリティーゾーンで置き換えレプリカが作成されます。

クラスター全体に障害が発生したため、データが失われ、すべての新しいノードがコールド起動されます。

置換先の各ノードと置換元のノードはエンドポイントが同じであるため、アプリケーションでエンドポイントを変更する必要はありません。

レプリケーショングループのエンドポイントの検索については、次のトピックを参照してください:

- [Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)
- [レプリケーショングループのエンドポイントの検索 \(AWS CLI\)](#)
- [レプリケーショングループのエンドポイントの検索 \(ElastiCache API\)](#)

耐障害性レベルを上げるために、プライマリノードとリードレプリカは別々のアベイラビリティーゾーンに作成することをお勧めします。

自動フェイルオーバーのテスト

自動フェイルオーバーを有効にする AWS CLI と、ElastiCache コンソール、および ElastiCache API を使用してテストできます。

テストを行う場合、以下の点に注意してください。

- このオペレーションを使用して、任意のローリング 24 時間で最大 15 個のシャード (ElastiCache API および `aws-elasticache` ではノードグループと呼ばれます AWS CLI) で自動フェイルオーバーをテストできます。
- 別のクラスターのシャード (API および CLI ではレプリケーショングループと呼ばれます) でこのオペレーションを呼び出す場合、同時に呼び出しを行うことができます。
- 場合によっては、同じ Redis (クラスターモードが有効) レプリケーショングループの複数の異なるシャードに対して、このオペレーションを複数回呼び出すことがあります。このような場合、後続の呼び出しを行う前に、最初のノードの置換が完了する必要があります。
- ノードの交換が完了したかどうかを判断するには、Amazon ElastiCache コンソール、AWS CLI または ElastiCache API を使用してイベントを確認します。自動フェイルオーバーに関連する次のイベントを検索します。ここでは、発生すると思われる順番にイベントを示します。
 1. レプリケーショングループメッセージ: Test Failover API called for node group <node-group-id>
 2. キャッシュクラスターメッセージ: Failover from primary node <primary-node-id> to replica node <node-id> completed
 3. レプリケーショングループメッセージ: Failover from primary node <primary-node-id> to replica node <node-id> completed
 4. キャッシュクラスターメッセージ: Recovering cache nodes <node-id>
 5. キャッシュクラスターメッセージ: Finished recovery for cache nodes <node-id>

詳細については、次を参照してください。

- 「ElastiCache ユーザーガイド」の「[ElastiCache イベントの表示](#)」
- 「[DescribeEvents](#) API リファレンス」の「ElastiCache」
- AWS CLI コマンドリファレンスの [describe-events](#)。
- この API は、ElastiCache フェイルオーバー時にアプリケーションの動作をテストするように設計されています。クラスターの問題に対処するためにフェイルオーバーを開始するための運用ツールとしては設計されていません。さらに、大規模な運用イベントなどの特定の条件下で AWS は、この API がブロックされる可能性があります。

トピック

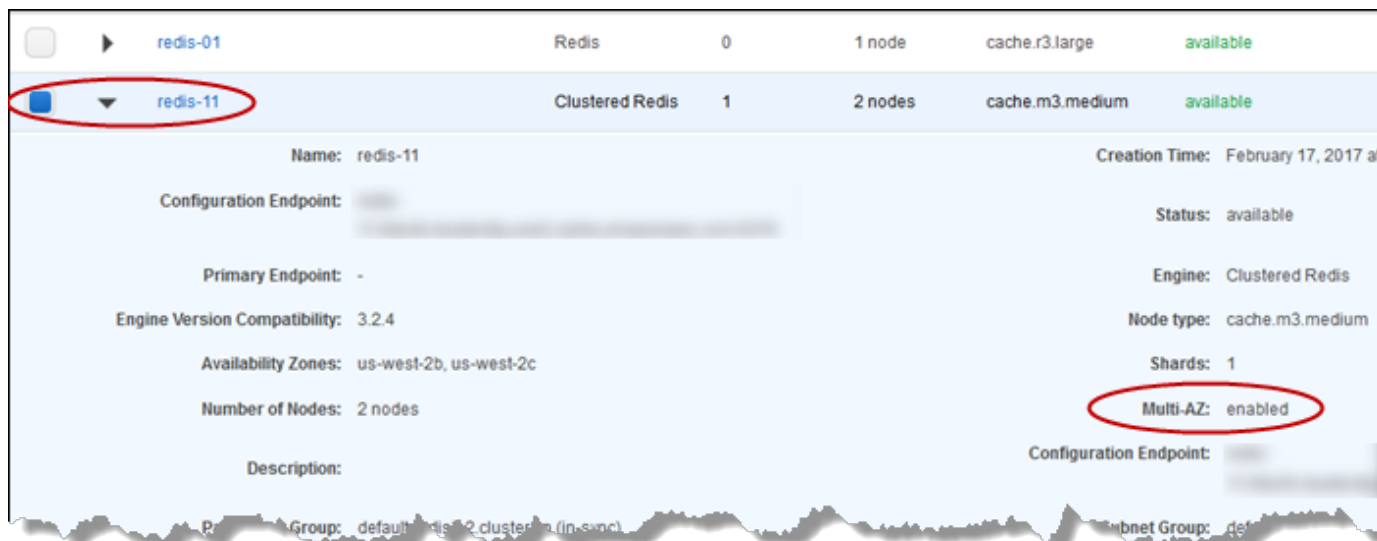
- [を使用した自動フェイルオーバーのテスト AWS Management Console](#)
- [を使用した自動フェイルオーバーのテスト AWS CLI](#)
- [ElastiCache API を使用した自動フェイルオーバーのテスト](#)

を使用した自動フェイルオーバーのテスト AWS Management Console

コンソールで自動フェイルオーバーをテストするには、次の手順に従います。

自動フェイルオーバーをテストするには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで [Redis] を選択します。
3. Redis クラスターの一覧で、テストするクラスターの名前の左にあるチェックボックスをオンにします。このクラスターには、少なくとも1つのリードレプリカノードが必要です。
4. Details エリアで、このクラスターでマルチ AZ が有効になっていることを確認します。クラスターでマルチ AZ が有効になっていない場合は、別のクラスターを選択するか、このクラスターを変更してマルチ AZ を有効にします。詳細については、「[の使用 AWS Management Console](#)」を参照してください。



5. Redis (クラスターモードが無効) の場合は、クラスターの名前を選択します。

Redis (クラスターモードが有効) の場合、次の手順を実行します。

- a. クラスターの名前を選択します。

- b. [Shards] ページで、フェイルオーバーをテストするシャード (API および CLI ではノードグループと呼ばれます) のシャード名を選択します。
6. [Nodes] ページで [Failover Primary] を選択します。
7. Continue を選択してプライマリをフェイルオーバーするか、Cancel を選択してプライマリノードへのフェイルオーバーをキャンセルします。

フェイルオーバープロセス中は、コンソールでノードのステータスが 使用可能 と継続して表示されます。フェイルオーバーテストの進捗状況を追跡するには、コンソールのナビゲーションペインから Events を選択します。Events タブで、フェイルオーバーの開始 Test Failover API called と完了 Recovery completed を示すイベントを監視します。

を使用した自動フェイルオーバーのテスト AWS CLI

AWS CLI オペレーション を使用して、マルチ AZ 対応クラスターで自動フェイルオーバーをテストできます test-failover。

パラメータ

- --replication-group-id – 必須。テストするレプリケーショングループ (コンソールではクラスター)。
- --node-group-id – 必須。自動フェイルオーバーをテストするノードグループの名前。24 時間周期で最大 15 個のノードグループをテストできます。

次の例では、 を使用して AWS CLI、Redis (クラスターモードが有効) クラスター のノードグループ redis00-0003 で自動フェイルオーバーをテストします redis00。

Example 自動フェイルオーバーをテストする

Linux、macOS、Unix の場合:

```
aws elasticache test-failover \  
  --replication-group-id redis00 \  
  --node-group-id redis00-0003
```

Windows の場合:

```
aws elasticache test-failover ^
```

```
--replication-group-id redis00 ^  
--node-group-id redis00-0003
```

上のコマンドによる出力は次のようになります。

```
{  
  "ReplicationGroup": {  
    "Status": "available",  
    "Description": "1 shard, 3 nodes (1 + 2 replicas)",  
    "NodeGroups": [  
      {  
        "Status": "available",  
        "NodeGroupMembers": [  
          {  
            "CurrentRole": "primary",  
            "PreferredAvailabilityZone": "us-west-2c",  
            "CacheNodeId": "0001",  
            "ReadEndpoint": {  
              "Port": 6379,  
              "Address":  
"redis1x3-001.7ekv3t.0001.usw2.cache.amazonaws.com"  
            },  
            "CacheClusterId": "redis1x3-001"  
          },  
          {  
            "CurrentRole": "replica",  
            "PreferredAvailabilityZone": "us-west-2a",  
            "CacheNodeId": "0001",  
            "ReadEndpoint": {  
              "Port": 6379,  
              "Address":  
"redis1x3-002.7ekv3t.0001.usw2.cache.amazonaws.com"  
            },  
            "CacheClusterId": "redis1x3-002"  
          },  
          {  
            "CurrentRole": "replica",  
            "PreferredAvailabilityZone": "us-west-2b",  
            "CacheNodeId": "0001",  
            "ReadEndpoint": {  
              "Port": 6379,  
              "Address":  
"redis1x3-003.7ekv3t.0001.usw2.cache.amazonaws.com"  
            }  
          }  
        ]  
      }  
    ]  
  }  
}
```



```
        },
        "CacheClusterId": "redis1x3-003"
    }
],
"NodeGroupId": "0001",
"PrimaryEndpoint": {
    "Port": 6379,
    "Address": "redis1x3.7ekv3t.ng.0001.usw2.cache.amazonaws.com"
}
}
],
"ClusterEnabled": false,
"ReplicationGroupId": "redis1x3",
"SnapshotRetentionLimit": 1,
"AutomaticFailover": "enabled",
"MultiAZ": "enabled",
"SnapshotWindow": "11:30-12:30",
"SnapshottingClusterId": "redis1x3-002",
"MemberClusters": [
    "redis1x3-001",
    "redis1x3-002",
    "redis1x3-003"
],
"CacheNodeType": "cache.m3.medium",
"DataTiering": "disabled",
"PendingModifiedValues": {}
}
}
```

フェイルオーバーの進行状況を追跡するには、AWS CLI `describe-events` オペレーションを使用します。

詳細については、次を参照してください。

- AWS CLI コマンドリファレンスの [test-failover](#)。
- AWS CLI コマンドリファレンスの [describe-events](#)。

ElastiCache API を使用した自動フェイルオーバーのテスト

ElastiCache API オペレーション を使用して、マルチ AZ が有効になっているクラスターで自動フェイルオーバーをテストできます `TestFailover`。

パラメータ

- `ReplicationGroupId` – 必須。テスト対象のレプリケーショングループ (コンソールではクラスター)。
- `NodeGroupId` – 必須。自動フェイルオーバーをテストする対象のノードグループの名前。24 時間周期で最大 15 個のノードグループをテストできます。

次の例では、レプリケーショングループ (コンソールではクラスター) `redis00` のノードグループ `redis00-0003` で、自動フェイルオーバーをテストします。

Example 自動フェイルオーバーのテスト

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=TestFailover  
&NodeGroupId=redis00-0003  
&ReplicationGroupId=redis00  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20140401T192317Z  
&X-Amz-Credential=<credential>
```

フェイルオーバーの進行状況を追跡するには、API `ElastiCache DescribeEvents` オペレーションを使用します。

詳細については、次を参照してください。

- [TestFailover](#) ElastiCache API リファレンスの
- [DescribeEvents](#) ElastiCache API リファレンスの

Redis のマルチ AZ の制限事項

Redis のマルチ AZ に関する次の制限事項に注意してください。

- マルチ AZ は、Redis バージョン 2.8.6 以降でサポートされます。
- Redis のマルチ AZ は、T1 ノードタイプではサポートされません。
- Redis レプリケーションは非同期で行われます。そのため、プライマリノードがレプリカにフェイルオーバーすると、レプリケーションの遅延のために少量のデータが失われる可能性があります。

プライマリに昇格するレプリカを選択すると、Redis ElastiCache ではレプリケーションラグが最も少ないレプリカが選択されます。言い換えると、最新のレプリカを選択します。これにより、失われるデータ量が最小限に抑えられます。レプリケーションの遅延が最短のレプリカは、障害が発生したプライマリノードと同じ、または異なるアベイラビリティーゾーンに存在できます。

- マルチ AZ と自動フェイルオーバーが無効なときにのみ、Redis (クラスターモードが無効) でリードレプリカを手動でプライマリに昇格させることができます。リードレプリカをプライマリに昇格させるには、以下のステップを実行します。
 1. クラスターでマルチ AZ を無効にします。
 2. クラスターで自動フェイルオーバーを無効にします。これを行うには、Redis コンソールを使用して、レプリケーショングループの [自動フェイルオーバー] チェックボックスをクリアします。これを行うには、ModifyReplicationGroupオペレーションを呼び出すfalseときに AutomaticFailoverEnabledプロパティを AWS CLI に設定します。
 3. リードレプリカをプライマリに昇格させます。
 4. マルチ AZ を再度有効にします。
- ElastiCache for Redis マルチ AZ と追加専用ファイル (AOF) は相互に排他的です。一方を有効にすると、他方を有効にすることはできません。
- アベイラビリティーゾーン全体の障害というまれなイベントにより、ノードの障害が発生することがあります。この場合、障害の発生したプライマリを置き換えるレプリカは、アベイラビリティーゾーンがバックアップされているときのみ作成されます。たとえば、AZ-a のプライマリおよび AZ-b および AZ-c のレプリカを持つレプリケーショングループを考えてみます。プライマリに障害が発生した場合、レプリケーションの遅延が最も小さい利用可能なレプリカをプライマリクラスターに昇格します。次に、は、AZ-a がバックアップされ、使用可能な場合にのみ、AZ-a (障害が発生したプライマリが配置されている) に新しいレプリカ ElastiCache を作成します。
- プライマリの再起動をお客様が開始した場合、自動フェイルオーバーはトリガーされません。他の再起動と障害は、自動フェイルオーバーをトリガーします。
- プライマリが再起動すると、オンラインに戻ったときにデータがクリアされます。リードレプリカがクリアされたプライマリクラスターを検出すると、データのコピーがクリアされるため、データ損失が発生します。

- リードレプリカが昇格されると、他のレプリカは新しいプライマリと同期されます。最初の同期後に、レプリカのコンテンツは削除され、新しいプライマリからデータが同期されます。この同期プロセスに伴って一時的に中断が発生し、その間はレプリカにアクセスできなくなります。また、この同期プロセスに伴ってレプリカとの同期中にプライマリで一時的にロードが増えます。この動作は Redis にネイティブであり、マルチ AZ に ElastiCache 固有ではありません。Redis の当該動作の詳細については、Redis ウェブサイトの「[のレプリケーション](#)」を参照してください。

⚠ Important

Redis バージョン 2.8.22 以降では、外部レプリカを作成できません。

2.8.22 より前の Redis バージョンでは、マルチ AZ が有効になっている for Redis クラスターに外部 Redis ElastiCache レプリカを接続しないことをお勧めします。サポートされていない設定では、ガフェイルオーバーとリカバリを適切に実行 ElastiCache できない問題が発生する可能性があります。外部 Redis レプリカを ElastiCache クラスターに接続するには、接続する前にマルチ AZ が有効になっていないことを確認してください。

同期とバックアップの実装方法

すべてのサポートされている Redis バージョンでは、プライマリノードとレプリカノード間でバックアップと同期がサポートされます。ただし、バックアップと同期の実装方法は Redis バージョンによって異なります。

Redis バージョン 2.8.22 以降

バージョン 2.8.22 以降の Redis レプリケーションでは、2 つの方法から選択します。詳細については、[Redis バージョン 2.8.22 以前](#) および [スナップショットおよび復元](#) を参照してください。

分岐なしプロセス中に書き込み負荷が高い場合は、クラスターへの書き込みを遅延させて、変更が蓄積しすぎて正常なスナップショットが妨げられないようにします。

Redis バージョン 2.8.22 以前

バージョン 2.8.22 以前の Redis のバックアップと同期は、3 つのステップで構成されるプロセスです。

1. バックグラウンドプロセスでは、分岐によりクラスターのデータがディスクにシリアル化されます。これは、特定の時点のスナップショットを作成します。
2. フォアグラウンドでは、クライアント出力バッファーに変更ログが蓄積されます。

Important

変更ログがクライアント出力バッファーのサイズを超えると、バックアップまたは同期が失敗します。詳細については、「[Redis スナップショットを作成するのに十分なメモリがあることの確認](#)」を参照してください。

3. 最後にキャッシュデータが送信され、変更ログがレプリカノードに転送されます。

Redis レプリケーショングループの作成

レプリカノードのあるクラスターを作成するには、以下のオプションがあります。1つは、プライマリノードとして使用するレプリカのあるクラスターに関連付けられていない Redis (クラスターモードが無効) クラスターがすでにある場合に適用されます。もう1つは、クラスターとリードレプリカのあるプライマリノードを作成する必要がある場合に適用されます。現時点では、Redis (クラスターモードが有効) クラスターを最初から作成する必要があります。

オプション 1: [利用可能な Redis \(クラスターモードが無効\) クラスターを使用したレプリケーショングループの作成](#)

このオプションは、既存の単一ノード Redis (クラスターモードが無効) クラスターを利用する場合に使用します。この既存ノードを、新しいクラスターのプライマリノードとして指定し、さらにクラスターに 1~5 個のリードレプリカを個別に追加します。既存のクラスターがアクティブの場合、リードレプリカは作成時にそのクラスターと同期されます。「[利用可能な Redis \(クラスターモードが無効\) クラスターを使用したレプリケーショングループの作成](#)」を参照してください。

Important

Redis (クラスターモードが有効) クラスターは、既存のクラスターを使用して作成できません。ElastiCache コンソールを使用して Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ) を作成するには、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。

オプション 2: [ゼロからの Redis レプリケーショングループの作成](#)

このオプションは、クラスターのプライマリノードとして使用可能な Redis (クラスターモードが無効) クラスターがまだない場合、または Redis (クラスターモードが有効) クラスターを作成する場合に使用します。「[ゼロからの Redis レプリケーショングループの作成](#)」を参照してください。

利用可能な Redis (クラスターモードが無効) クラスターを使用したレプリケーショングループの作成

使用可能なクラスターは、単一ノードの Redis クラスターです。現時点では、Redis (クラスターモードが有効) では使用可能な単一ノードのクラスターを使用して、レプリカを持つクラスターを作成することはできません。Redis (クラスターモードが有効) クラスターを作成する場合は、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。

次の手順は、単一ノードの Redis (クラスターモードが無効) クラスターがある場合に限り使用できます。このノードは新しいクラスターのプライマリノードになります。新しいクラスターのプライマリとして使用できる Redis (クラスターモードが無効) クラスターがない場合は、「[ゼロからの Redis レプリケーショングループの作成](#)」を参照してください。

利用可能な Redis クラスターを使用したレプリケーショングループの作成 (コンソール)

トピック「[AWS Management Console を使用する場合](#)」を参照してください。

使用可能な Redis キャッシュクラスターを使用したレプリケーショングループの作成 (AWS CLI)

AWS CLIを使う場合、使用可能な Redis キャッシュクラスターをプライマリとして、リードレプリカを持つレプリケーショングループを作成するには 2 つのステップがあります。

を使用する場合、クラスターのプライマリノードとして使用可能なスタンドアロンノード --primary-cluster-id と、CLI コマンドを使用してクラスターに必要なノードの数を指定するレプリケーショングループ AWS CLI を作成します create-replication-group。以下のパラメータを含めます。

--replication-group-id

作成するレプリケーショングループの名前。このパラメータの値が追加されたノードの名前の基礎として使用され、3 桁の連番が --replication-group-id の末尾に追加されます。例えば sample-repl-group-001 です。

Redis (クラスターモードが無効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

--replication-group-description

レプリケーショングループの説明。

--num-node-groups

このクラスターに必要なノードの数。この値はプライマリノードを含みます。このパラメータの最大値は 6 です。

--primary-cluster-id

このレプリケーショングループのプライマリノードにする、使用可能な Redis (クラスターモードが無効) クラスターの名前。

次のコマンドは、レプリケーショングループ `sample-repl-group` を作成します。レプリケーショングループのプライマリノードとして使用できる Redis (クラスターモードが無効) クラスター `redis01` を使用します。リードレプリカとなる 2 つの新しいノードを作成します。`redis01` の設定 (つまり、パラメータグループ、セキュリティグループ、ノードタイプ、エンジンバージョンなど) は、レプリケーショングループ内のすべてのノードに適用されます。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id sample-repl-group \  
  --replication-group-description "demo cluster with replicas" \  
  --num-cache-clusters 3 \  
  --primary-cluster-id redis01
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id sample-repl-group ^  
  --replication-group-description "demo cluster with replicas" ^  
  --num-cache-clusters 3 ^  
  --primary-cluster-id redis01
```

使用する可能性のある追加情報とパラメータについては、AWS CLI 「」トピックを参照してください [create-replication-group](#)。

次に、リードレプリカをレプリケーショングループに追加します。

レプリケーショングループの作成後に、`create-cache-cluster` コマンドを使用して、そのグループに 1 ～ 5 個のリードレプリカを追加します。その際に、以下のパラメータを必ず含めます。

`--cache-cluster-id`

レプリケーショングループに追加するクラスターの名前。

クラスターの命名に関する制約は次のとおりです。

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

`--replication-group-id`

このキャッシュクラスターに追加するレプリケーショングループの名前。

レプリケーショングループに追加するそれぞれのリードレプリカで、このコマンドを `--cache-cluster-id` パラメータの値のみを変更して繰り返します。

Note

レプリケーショングループに追加できるリードレプリカの数 は 5 個までです。すでに 5 個のリードレプリカを持つレプリケーショングループに別のリードレプリカを追加しようとすると、オペレーションが失敗します。

次のコードは、リードレプリカ `my-replica01` をレプリケーショングループ `sample-repl-group` に追加します。プライマリクラスター-パラメータグループ、セキュリティグループ、ノードタイプなどの設定は、レプリケーショングループに追加されるノードに適用されます。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-cluster \  
  --cache-cluster-id my-replica01 \  
  --replication-group-id sample-repl-group
```

Windows の場合:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-replica01 ^
--replication-group-id sample-repl-group
```

このコマンドの出力は次のようになります。

```
{
  "ReplicationGroup": {
    "Status": "creating",
    "Description": "demo cluster with replicas",
    "ClusterEnabled": false,
    "ReplicationGroupId": "sample-repl-group",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "disabled",
    "SnapshotWindow": "00:00-01:00",
    "SnapshottingClusterId": "redis01",
    "MemberClusters": [
      "sample-repl-group-001",
      "sample-repl-group-002",
      "redis01"
    ],
    "CacheNodeType": "cache.m4.large",
    "DataTiering": "disabled",
    "PendingModifiedValues": {}
  }
}
```

詳細については、以下の AWS CLI トピックを参照してください。

- [create-replication-group](#)
- [modify-replication-group](#)

スタンドアロン Redis (クラスターモードが無効) クラスターへのレプリカの追加 (ElastiCache API)

ElastiCache API を使用する場合は、CLI コマンド を使用して、使用可能なスタンドアロンノードをクラスターのプライマリノードとして指定 `PrimaryClusterId` し、クラスターに必要なノードの数を指定するレプリケーショングループを作成します `CreateReplicationGroup`。以下のパラメータを含めます。

ReplicationGroupId

作成するレプリケーショングループの名前。このパラメータの値が追加されたノードの名前の基礎として使用され、3桁の連番が ReplicationGroupId の末尾に追加されます。例えば sample-repl-group-001 です。

Redis (クラスターモードが無効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

ReplicationGroup説明

レプリカを持つクラスターの説明。

NumCacheクラスター

このクラスターに必要なノードの数。この値はプライマリノードを含みます。このパラメータの最大値は 6 です。

PrimaryClusterID

このクラスター内のプライマリノードにする Redis (クラスターモード無効) クラスターの名前。

次のコマンドは、レプリカを持つクラスター sample-repl-group を作成します。レプリケーショングループのプライマリノードとして使用できる Redis (クラスターモードが無効) クラスター redis01 を使用します。リードレプリカとなる 2 つの新しいノードを作成します。redis01 の設定 (つまり、パラメータグループ、セキュリティグループ、ノードタイプ、エンジンバージョンなど) は、レプリケーショングループ内のすべてのノードに適用されます。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=CreateReplicationGroup  
&Engine=redis  
&EngineVersion=6.0  
&ReplicationGroupDescription=Demo%20cluster%20with%20replicas  
&ReplicationGroupId=sample-repl-group  
&PrimaryClusterId=redis01  
&Version=2015-02-02  
&SignatureVersion=4
```

```
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

詳細については、APL ElastiCache トピックを参照してください。

- [CreateReplicationグループ](#)
- [ModifyReplicationグループ](#)

次に、リードレプリカをレプリケーショングループに追加します。

レプリケーショングループの作成後に、CreateCacheCluster オペレーションを使用して、そのグループに 1～5 個のリードレプリカを追加します。その際に、以下のパラメータを必ず含めます。

CacheClusterID

レプリケーショングループに追加するクラスターの名前。

クラスターの命名に関する制約は次のとおりです。

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

ReplicationGroupID

このキャッシュクラスターに追加するレプリケーショングループの名前。

レプリケーショングループに追加するリードレプリカごとに、このオペレーションを繰り返します。その際に、CacheClusterId パラメータの値のみを変更します。

次のコードは、リードレプリカ myReplica01 をレプリケーショングループ myReplGroup に追加します。プライマリクラスター-パラメータグループ、セキュリティグループ、ノードタイプなどの設定です。はレプリケーショングループに追加されると、ノードに適用されます。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateCacheCluster
```

```
&CacheClusterId=myReplica01
&ReplicationGroupId=myReplGroup
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2015-02-02
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Credential=[your-access-key-id]/20150202/us-west-2/elasticache/aws4_request
&X-Amz-Date=20150202T170651Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=[signature-value]
```

使用する可能性のある追加情報とパラメータについては、ElastiCache「API トピック」を参照してください [CreateCacheCluster](#)。

ゼロからの Redis レプリケーショングループの作成

次に、既存の Redis クラスターをプライマリとして使用せずに、Redis レプリケーショングループを作成する方法について説明します。Redis (クラスターモードが無効) または Redis (クラスターモードが有効) レプリケーショングループを最初から作成するには、ElastiCache コンソール、AWS CLI、または ElastiCache API を使用します。

続行する前に、Redis (クラスターモードが無効) レプリケーショングループを作成するのか、Redis (クラスターモードが有効) レプリケーショングループを作成するのかを決定します。決定のガイダンスについては、「[レプリケーション: Redis \(クラスターモードが無効\) 対 Redis \(クラスターモードが有効\)](#)」を参照してください。

トピック

- [Redis \(クラスターモードが無効\) レプリケーショングループを最初から作成する](#)
- [Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する](#)

Redis (クラスターモードが無効) レプリケーショングループを最初から作成する

コンソール、AWS CLI または ElastiCache API を使用して ElastiCache、Redis (クラスターモードが無効) レプリケーショングループを最初から作成できます。Redis (クラスターモードが無効) レプリケーショングループには常に 1 つのノードグループ、プライマリクラスター、および最大 5 個のリードレプリカがあります。Redis (クラスターモードが無効) レプリケーショングループは、データのパーティション化をサポートしていません。

Note

ノード/シャード制限は、クラスターあたり 500 まで増やすことができます。この制限の拡大をリクエストするには、「[AWS サービスの制限](#)」を参照し、リクエストにインスタンスタイプを含めます。

Redis (クラスターモードが無効) レプリケーショングループを最初から作成するには、次のいずれかの方法に従います。

Redis (クラスターモードが無効) レプリケーショングループを最初から作成する (AWS CLI)

以下の手順では、AWS CLI を使用して Redis (クラスターモードが無効) レプリケーショングループを作成します。

Redis (クラスターモードが無効) レプリケーショングループを最初から作成する場合は、`create-replication-group` コマンドを AWS CLI 1 回呼び出すだけで、レプリケーショングループとすべてのノードを作成します。以下のパラメータを含めます。

`--replication-group-id`

作成するレプリケーショングループの名前。

Redis (クラスターモードが無効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

`--replication-group-description`

レプリケーショングループの説明。

--num-cache-clusters

このレプリケーションのグループ、プライマリおよびリードレプリカ全体で作成するノードの数。

マルチ AZ を有効にした場合 (--automatic-failover-enabled)、--num-cache-clusters の値は 2 以上であることが必要です。

--cache-node-type

レプリケーショングループの各ノードのノードタイプ。

ElastiCache では、次のノードタイプがサポートされています。一般に、現行世代のタイプは、以前の世代の同等タイプと比較した場合、メモリが多く処理能力が高くなっています。

各ノードタイプのパフォーマンスの詳細の詳細については、「[Amazon EC2 インスタンスタイプ](#)」を参照してください。

--data-tiering-enabled

r6gd ノードタイプを使用している場合は、このパラメータを設定します。データ階層化の必要がない場合は、--no-data-tiering-enabled を設定します。詳細については、「[データ階層化](#)」を参照してください。

--cache-parameter-group

エンジンバージョンに対応するパラメータグループを指定します。Redis 3.2.4 以降を実行している場合は、default.redis3.2 パラメータグループ、または default.redis3.2 から派生したパラメータグループを指定して Redis (クラスターモードが無効) レプリケーショングループを指定します。詳細については、「[Redis 固有のパラメータ](#)」を参照してください。

--network-type

ipv4、ipv6 または dual-stack です。デュアルスタックを選択する場合は、--IpDiscovery パラメータを ipv4 または ipv6 に設定する必要があります。

--engine

redis

--engine-version

最も豊富な機能のセットを利用するには、最新バージョンのエンジンを選択します。

ノードの名前は、レプリケーショングループ名の後に「-00#」を追加することで決定されます。たとえば、レプリケーショングループ名 `myReplGroup` を使用すると、プライマリの名前は `myReplGroup-002` となり、リードレプリカの名前は `myReplGroup-001` から `myReplGroup-006` となります。

このレプリケーショングループで転送時または保管時の暗号化を有効にする場合は、`--transit-encryption-enabled` パラメータと `--at-rest-encryption-enabled` パラメータの一方または両方を追加し、次の条件を満たす必要があります。

- レプリケーショングループは Redis バージョン 3.2.6 または 4.0.10.
- レプリケーショングループは Amazon VPC で作成されている必要があります。
- パラメータ `--cache-subnet-group` も含める必要があります。
- このレプリケーショングループに対するオペレーションを実行するために必要な AUTH トークン (パスワード) 用に顧客が指定した文字列値にパラメータ `--auth-token` も含める必要があります。

次のオペレーションでは、3 つのノード、1 つのプライマリ、2 つのレプリカを持つ Redis (クラスターモードが無効) レプリケーショングループ `sample-repl-group` を作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id sample-repl-group \  
  --replication-group-description "Demo cluster with replicas" \  
  --num-cache-clusters 3 \  
  --cache-node-type cache.m4.large \  
  --engine redis
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id sample-repl-group ^  
  --replication-group-description "Demo cluster with replicas" ^  
  --num-cache-clusters 3 ^  
  --cache-node-type cache.m4.large ^  
  --engine redis
```

このコマンドによる出力は次のようになります。

```
{
  "ReplicationGroup": {
    "Status": "creating",
    "Description": "Demo cluster with replicas",
    "ClusterEnabled": false,
    "ReplicationGroupId": "sample-repl-group",
    "SnapshotRetentionLimit": 0,
    "AutomaticFailover": "disabled",
    "SnapshotWindow": "01:30-02:30",
    "MemberClusters": [
      "sample-repl-group-001",
      "sample-repl-group-002",
      "sample-repl-group-003"
    ],
    "CacheNodeType": "cache.m4.large",
    "DataTiering": "disabled",
    "PendingModifiedValues": {}
  }
}
```

使用する可能性のある追加情報とパラメータについては、[「create-replication-group」](#)の AWS CLI トピックを参照してください。

Redis (クラスターモードが無効) レプリケーショングループを最初から作成する (ElastiCache API)

次の手順では、ElastiCache API を使用して Redis (クラスターモードが無効) レプリケーショングループを作成します。

Redis (クラスターモードが無効) レプリケーショングループを最初から作成する場合は、ElastiCache API CreateReplicationGroup オペレーションを 1 回呼び出すだけで、レプリケーショングループとそのすべてのノードを作成します。以下のパラメータを含めます。

ReplicationGroupID

作成するレプリケーショングループの名前。

Redis (クラスターモードが有効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

- ハイフンで終わることはできません。

ReplicationGroup説明

レプリケーショングループの説明。

NumCacheクラスター

このレプリケーションのグループ、プライマリおよびリードレプリカ全体で作成するノードの総数。

マルチ AZ を有効にした場合 (AutomaticFailoverEnabled=true)、NumCacheClusters の値は 2 以上であることが必要です。

CacheNodeタイプ

レプリケーショングループの各ノードのノードタイプ。

ElastiCache では、次のノードタイプがサポートされています。一般に、現行世代のタイプは、以前の世代の同等タイプと比較した場合、メモリが多く処理能力が高くなっています。

各ノードタイプのパフォーマンスの詳細の詳細については、「[Amazon EC2 インスタンスタイプ](#)」を参照してください。

--data-tiering-enabled

r6gd ノードタイプを使用している場合は、このパラメータを設定します。データ階層化の必要がない場合は、--no-data-tiering-enabled を設定します。詳細については、「[データ階層化](#)」を参照してください。

CacheParameterグループ

エンジンバージョンに対応するパラメータグループを指定します。Redis 3.2.4 以降を実行している場合は、default.redis3.2 パラメータグループ、または default.redis3.2 から派生したパラメータグループを指定して Redis (クラスターモードが無効) レプリケーショングループを指定します。詳細については、「[Redis 固有のパラメータ](#)」を参照してください。

--network-type

ipv4、ipv または dual-stack です。デュアルスタックを選択する場合は、--IpDiscovery パラメータを ipv4 または ipv6 に設定する必要があります。

エンジン

redis

EngineVersion

6.0

ノードの名前は、レプリケーショングループ名の後に「-00#」を追加することで決定されます。たとえば、レプリケーショングループ名 myReplGroup を使用すると、プライマリの名前は myReplGroup-002 となり、リードレプリカの名前は myReplGroup-001 から myReplGroup-006 となります。

このレプリケーショングループで転送時または保管時の暗号化を有効にする場合は、TransitEncryptionEnabled=true パラメータと AtRestEncryptionEnabled=true パラメータの一方または両方を追加し、次の条件を満たす必要があります。

- レプリケーショングループは Redis バージョン 3.2.6 または 4.0.10.
- レプリケーショングループは Amazon VPC で作成されている必要があります。
- パラメータ CacheSubnetGroup も含める必要があります。
- このレプリケーショングループに対するオペレーションを実行するために必要な AUTH トークン (パスワード) 用に顧客が指定した文字列値にパラメータ AuthToken も含める必要があります。

次のオペレーションでは、3 つのノード、1 つのプライマリ、2 つのレプリカを持つ Redis (クラスターモードが無効) レプリケーショングループ myReplGroup を作成します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=CreateReplicationGroup  
&CacheNodeType=cache.m4.large  
&CacheParameterGroup=default.redis6.x  
&Engine=redis  
&EngineVersion=6.0  
&NumCacheClusters=3  
&ReplicationGroupDescription=test%20group  
&ReplicationGroupId=myReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

使用する可能性のある追加情報とパラメータについては、ElastiCache 「API トピック」を参照してください [CreateReplicationGroup](#)。

Redis (クラスターモードが有効) レプリケーショングループを最初から作成する

ElastiCache コンソール、AWS CLI または ElastiCache API を使用して、Redis (クラスターモードが有効) クラスター (API/CLI: レプリケーショングループ) を作成できます。Redis (クラスターモードが有効) レプリケーショングループは、1~500 個のシャード (API/CLI: ノードグループ) で構成され、各シャードには、1つのプライマリクラスターと、最大 5 個のリードレプリカが含まれます。シャードの数が多くレプリカの数が少ないクラスターを作成できます。クラスターあたり最大 90 ノードです。このクラスター設定は、シャード 90 個およびレプリカ 0 個からシャード 15 個およびレプリカ 5 個 (許容されるレプリカの最大数) までです。

Redis エンジンのバージョンが 5.0.6 以上の場合、ノードまたはシャードの制限は、クラスターごとに最大 500 個に増やすことができます。例えば、83 個のシャード (シャードごとに 1つのプライマリと 5レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。詳細については、「[サブネットグループの作成](#)」を参照してください。

5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

この制限の拡大をリクエストするには、「[AWS のサービスの制限](#)」を参照し、制限タイプとして [Nodes per cluster per instance type (インスタンスタイプごとのクラスターあたりのノード)] を選択します。

Redis (クラスターモードが有効) でのクラスターの作成

- [Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)
- [Redis \(クラスターモードが有効\) でのレプリケーショングループを最初から作成する \(ElastiCache API\)](#)

Redis (クラスターモードが有効) クラスターの作成 (コンソール)

Redis (クラスターモード対応) クラスターを作成するには、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。クラスターモード ([クラスターモードが有効 (スケールアウト)]) を必ず有効にし、それぞれに少なくとも 2 つのシャードと 1 つのレプリカノードを指定します。

Redis (クラスターモードが有効) レプリケーショングループを最初から作成する (AWS CLI)

以下の手順では、AWS CLIを使用して Redis (クラスターモードが有効) レプリケーショングループを作成します。

Redis (クラスターモードが有効) レプリケーショングループを最初から作成する場合は、`create-replication-group` コマンドを AWS CLI 1 回呼び出すだけで、レプリケーショングループとそのすべてのノードを作成します。以下のパラメータを含めます。

`--replication-group-id`

作成するレプリケーショングループの名前。

Redis (クラスターモードが有効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

`--replication-group-description`

レプリケーショングループの説明。

`--cache-node-type`

レプリケーショングループの各ノードのノードタイプ。

ElastiCache では、次のノードタイプがサポートされています。一般に、現行世代のタイプは、以前の世代の同等タイプと比較した場合、メモリが多く処理能力が高くなっています。

各ノードタイプのパフォーマンスの詳細の詳細については、「[Amazon EC2 インスタンスタイプ](#)」を参照してください。

`--data-tiering-enabled`

r6gd ノードタイプを使用している場合は、このパラメータを設定します。データ階層化の必要がない場合は、`--no-data-tiering-enabled` を設定します。詳細については、「[データ階層化](#)」を参照してください。

`--cache-parameter-group`

`default.redis6.x.cluster.on` パラメータグループまたは `default.redis6.x.cluster.on` から派生したパラメータグループを指定して、Redis (クラ

スターモードが有効) レプリケーショングループを作成します。詳細については、「[Redis 6.x パラメータの変更](#)」を参照してください。

--engine


redis

--engine-version

3.2.4

--num-node-groups

このレプリケーショングループのノードグループの数。有効な値は 1~500 です。

 Note

ノード/シャード制限は、クラスターあたり 500 まで増やすことができます。この制限の拡大をリクエストするには、「[AWS サービスの制限](#)」を参照し、制限タイプ「インスタンスタイプごとのクラスターあたりのノード」を選択します。

--replicas-per-node-group

各ノードグループのレプリカノードの数。有効な値は 0~5 です。

--network-type

ipv4、ipv または dual-stack です。デュアルスタックを選択する場合は、--IpDiscovery パラメータを ipv4 または ipv6 に設定する必要があります。

このレプリケーショングループで転送時または保管時の暗号化を有効にする場合は、--transit-encryption-enabled パラメータと --at-rest-encryption-enabled パラメータの一方または両方を追加し、次の条件を満たす必要があります。

- レプリケーショングループは Redis バージョン 3.2.6 または 4.0.10.
- レプリケーショングループは Amazon VPC で作成されている必要があります。
- パラメータ --cache-subnet-group も含める必要があります。
- このレプリケーショングループに対するオペレーションを実行するために必要な AUTH トークン (パスワード) 用に顧客が指定した文字列値にパラメータ --auth-token も含める必要があります。

次のオペレーションでは、3つのノードグループおよびシャード (--num-node-groups) を持つ Redis (クラスターモードが有効) レプリケーショングループ `sample-repl-group` を作成します。各レプリケーショングループに3つのノード、1つのプライマリ、2つのリードレプリカ (--replicas-per-node-group) が含まれます。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id sample-repl-group \  
  --replication-group-description "Demo cluster with replicas" \  
  --num-node-groups 3 \  
  --replicas-per-node-group 2 \  
  --cache-node-type cache.m4.large \  
  --engine redis \  
  --security-group-ids SECURITY_GROUP_ID \  
  --cache-subnet-group-name SUBNET_GROUP_NAME>
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id sample-repl-group ^  
  --replication-group-description "Demo cluster with replicas" ^  
  --num-node-groups 3 ^  
  --replicas-per-node-group 2 ^  
  --cache-node-type cache.m4.large ^  
  --engine redis ^  
  --security-group-ids SECURITY_GROUP_ID ^  
  --cache-subnet-group-name SUBNET_GROUP_NAME>
```

前述のコマンドは、次の出力を生成します。

```
{  
  "ReplicationGroup": {  
    "Status": "creating",  
    "Description": "Demo cluster with replicas",  
    "ReplicationGroupId": "sample-repl-group",  
    "SnapshotRetentionLimit": 0,  
    "AutomaticFailover": "enabled",  
    "SnapshotWindow": "05:30-06:30",  
    "MemberClusters": [  

```



```
        "sample-repl-group-0001-001",
        "sample-repl-group-0001-002",
        "sample-repl-group-0001-003",
        "sample-repl-group-0002-001",
        "sample-repl-group-0002-002",
        "sample-repl-group-0002-003",
        "sample-repl-group-0003-001",
        "sample-repl-group-0003-002",
        "sample-repl-group-0003-003"
    ],
    "PendingModifiedValues": {}
}
}
```

Redis (クラスターモードが有効) レプリケーショングループを最初から作成する際、次の例に示すように `--node-group-configuration` パラメータを使用してクラスター内の各シャードを設定することで、2つのノードグループ (コンソール: シャード) を設定できます。1つめのシャードは、2つのノード、1つのプライマリ、1つのリードレプリカで構成されます。2つめのシャードは、3つのノード、1つのプライマリ、2つのリードレプリカで構成されます。

`--node-group-configuration`

各ノードグループの設定。 `--node-group-configuration` パラメータは次のフィールドで構成されます。

- `PrimaryAvailabilityZone` – このノードグループのプライマリノードがあるアベイラビリティゾーン。このパラメータを省略すると、プライマリノード ElastiCache のアベイラビリティゾーンが選択されます。

例: `us-west-2a`。

- `ReplicaAvailabilityZones` – リードレプリカがあるアベイラビリティゾーンのカンマ区切りリスト。このリストのアベイラビリティゾーンの数、`ReplicaCount` の値と一致する必要があります。このパラメータを省略すると、はレプリカノードのアベイラビリティゾーン ElastiCache を選択します。

例: `"us-west-2a,us-west-2b,us-west-2c"`

- `ReplicaCount` – このノードグループのレプリカノードの数。
- `Slots` – 対象ノードグループのキースペースを指定する文字列。この文字列は次の形式になります。 `startKey-endKey` このパラメータを省略すると、はノードグループ間でキーを均等 ElastiCache に割り当てます。

例: "0-4999"

次のオペレーションでは、2つのノードグループとシャード (--num-node-groups) を持つ Redis (クラスターモードが有効) レプリケーショングループ new-group を作成します。前の例とは異なり、各ノードグループは、その他のノードグループ (--node-group-configuration) とは異なった構成になります。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id new-group \  
  --replication-group-description "Sharded replication group" \  
  --engine redis \  
  --snapshot-retention-limit 8 \  
  --cache-node-type cache.m4.medium \  
  --num-node-groups 2 \  
  --node-group-configuration \  
    "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-east-1c',ReplicaAvailabilityZones='us-east-1b'" \  
    "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id new-group ^  
  --replication-group-description "Sharded replication group" ^  
  --engine redis ^  
  --snapshot-retention-limit 8 ^  
  --cache-node-type cache.m4.medium ^  
  --num-node-groups 2 ^  
  --node-group-configuration \  
    "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-east-1c',ReplicaAvailabilityZones='us-east-1b'" \  
    "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

前述のオペレーションは、次の出力を生成します。

```
{
  "ReplicationGroup": {
    "Status": "creating",
    "Description": "Sharded replication group",
    "ReplicationGroupId": "rc-rg",
    "SnapshotRetentionLimit": 8,
    "AutomaticFailover": "enabled",
    "SnapshotWindow": "10:00-11:00",
    "MemberClusters": [
      "rc-rg-0001-001",
      "rc-rg-0001-002",
      "rc-rg-0002-001",
      "rc-rg-0002-002",
      "rc-rg-0002-003"
    ],
    "PendingModifiedValues": {}
  }
}
```

使用する可能性のある追加情報とパラメータについては、AWS CLI 「」トピックを参照してください [create-replication-group](#)。

Redis (クラスターモードが有効) でのレプリケーショングループを最初から作成する (ElastiCache API)

次の手順では、ElastiCache API を使用して Redis (クラスターモードが有効) レプリケーショングループを作成します。

Redis (クラスターモードが有効) レプリケーショングループを最初から作成する場合は、ElastiCache API CreateReplicationGroup オペレーションを 1 回呼び出すだけで、レプリケーショングループとそのすべてのノードを作成します。以下のパラメータを含めます。

ReplicationGroupId

作成するレプリケーショングループの名前。

Redis (クラスターモードが有効) レプリケーショングループの命名に関する制約は、次のとおりです。

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

- ハイフンで終わることはできません。

ReplicationGroup説明

レプリケーショングループの説明。

NumNodeグループ

このレプリケーショングループで作成するノードグループの数。有効な値は 1~500 です。

ReplicasPerNodeGroup

各ノードグループのレプリカノードの数。有効な値は 1~5 です。

NodeGroup設定

各ノードグループの設定。NodeGroupConfiguration パラメータは次のフィールドで構成されます。

- PrimaryAvailabilityZone – このノードグループのプライマリノードがあるアベイラビリティゾーン。このパラメータを省略すると、プライマリノード ElastiCache のアベイラビリティゾーンが選択されます。

例: us-west-2a。

- ReplicaAvailabilityZones – リードレプリカがあるアベイラビリティゾーンのリスト。このリストのアベイラビリティゾーンの数は、ReplicaCount の値と一致する必要があります。このパラメータを省略すると、はレプリカノードのアベイラビリティゾーン ElastiCache を選択します。
- ReplicaCount – このノードグループのレプリカノードの数。
- Slots – 対象ノードグループのキースペースを指定する文字列。この文字列は次の形式になります。startKey-endKeyこのパラメータを省略すると、はノードグループ間でキーを均等 ElastiCache に割り当てます。

例: "0-4999"

CacheNodeタイプ

レプリケーショングループの各ノードのノードタイプ。

ElastiCache では、次のノードタイプがサポートされています。一般に、現行世代のタイプは、以前の世代の同等タイプと比較した場合、メモリが多く処理能力が高くなっています。

各ノードタイプのパフォーマンスの詳細の詳細については、「[Amazon EC2 インスタスタ](#)
[イプ](#)」を参照してください。

--data-tiering-enabled

r6gd ノードタイプを使用している場合は、このパラメータを設定します。データ階層化の必要がない場合は、--no-data-tiering-enabled を設定します。詳細については、「[データ階層化](#)」を参照してください。

CacheParameterグループ

default.redis6.x.cluster.on パラメータグループまたは default.redis6.x.cluster.on から派生したパラメータグループを指定して、Redis (クラスターモードが有効) レプリケーショングループを作成します。詳細については、「[Redis 6.x パラメータの変更](#)」を参照してください。

--network-type

ipv4、ipv または dual-stack です。デュアルスタックを選択する場合は、--IpDiscovery パラメータを ipv4 または ipv6 に設定する必要があります。

エンジン

redis

EngineVersion

6.0

このレプリケーショングループで転送時または保管時の暗号化を有効にする場合は、TransitEncryptionEnabled=true パラメータと AtRestEncryptionEnabled=true パラメータの一方または両方を追加し、次の条件を満たす必要があります。

- レプリケーショングループは Redis バージョン 3.2.6 または 4.0.10.
- レプリケーショングループは Amazon VPC で作成されている必要があります。
- パラメータ CacheSubnetGroup も含める必要があります。
- このレプリケーショングループに対するオペレーションを実行するために必要な AUTH トークン (パスワード) 用に顧客が指定した文字列値にパラメータ AuthToken も含める必要があります。

読みやすくするために、改行が追加されています。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=CreateReplicationGroup  
&CacheNodeType=cache.m4.large  
&CacheParameterGroup=default.redis6.xcluster.on  
&Engine=redis  
&EngineVersion=6.0  
&NumNodeGroups=3  
&ReplicasPerNodeGroup=2  
&ReplicationGroupDescription=test%20group  
&ReplicationGroupId=myReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

使用する可能性のある追加情報とパラメータについては、ElastiCache「API トピック」を参照してください [CreateReplicationGroup](#)。

レプリケーショングループの詳細の表示

レプリケーショングループの詳細を表示すると便利な場合があります。ElastiCache コンソール、AWS CLI for ElastiCache、またはElastiCache API を使用できます。コンソールプロセスは、Redis (クラスターモードが無効) と Redis (クラスターモードが有効) では異なります。

レプリケーショングループの詳細の表示

- [レプリカがある Redis \(クラスターモード無効\) の詳細の表示](#)
 - [Redis \(クラスターモードが無効\) レプリケーショングループの詳細の表示 \(コンソール\)](#)
 - [Redis \(クラスターモードが無効\) レプリケーショングループの詳細の表示 \(AWS CLI\)](#)
 - [Redis \(クラスターモードが無効\) レプリケーショングループ \(ElastiCache API\) の詳細の表示](#)
- [レプリケーショングループの詳細の表示: Redis \(クラスターモードが有効\)](#)
 - [Redis \(クラスターモードが有効\) クラスターの詳細の表示 \(コンソール\)](#)
 - [Redis \(クラスターモードが有効\) クラスターの詳細の表示 \(AWS CLI\)](#)
 - [Redis \(クラスターモードが有効\) クラスター \(ElastiCache API\) の詳細の表示](#)
- [レプリケーショングループの詳細の表示 \(AWS CLI\)](#)
- [レプリケーショングループの詳細の表示 \(ElastiCache API\)](#)

レプリカがある Redis (クラスターモード無効) の詳細の表示

ElastiCache コンソール、の、ElastiCache または ElastiCache API を使用して、レプリカ (API/CLI: レプリケーショングループ) を持つ Redis (クラスターモードが無効) AWS CLI クラスターの詳細を表示できます。

Redis (クラスターモードが無効) クラスターの詳細の表示

- [Redis \(クラスターモードが無効\) レプリケーショングループの詳細の表示 \(コンソール\)](#)
- [Redis \(クラスターモードが無効\) レプリケーショングループの詳細の表示 \(AWS CLI\)](#)
- [Redis \(クラスターモードが無効\) レプリケーショングループ \(ElastiCache API\) の詳細の表示](#)

Redis (クラスターモードが無効) レプリケーショングループの詳細の表示 (コンソール)

ElastiCache コンソールを使用してレプリカを持つ Redis (クラスターモードが無効) クラスターの詳細を表示するには、「」トピックを参照してください [Redis \(クラスターモードが無効\) クラスターの詳細の表示 \(コンソール\)](#)。

Redis (クラスターモードが無効) レプリケーショングループの詳細の表示 (AWS CLI)

Redis (クラスターモードが無効) レプリケーショングループの詳細を表示する AWS CLI 例については、「」を参照してください [レプリケーショングループの詳細の表示 \(AWS CLI\)](#)。

Redis (クラスターモードが無効) レプリケーショングループ (ElastiCache API) の詳細の表示

Redis (クラスターモードが無効) レプリケーショングループの詳細を表示する ElastiCache API の例については、「」を参照してください [レプリケーショングループの詳細の表示 \(ElastiCache API\)](#)。

レプリケーショングループの詳細の表示: Redis (クラスターモードが有効)

Redis (クラスターモードが有効) クラスターの詳細の表示 (コンソール)

ElastiCache コンソールを使用して Redis (クラスターモードが有効) クラスターの詳細を表示するには、「」を参照してください [Redis \(クラスターモードが有効\) クラスターの詳細の表示 \(コンソール\)](#)。

Redis (クラスターモードが有効) クラスターの詳細の表示 (AWS CLI)

Redis (クラスターモードが有効) レプリケーショングループの詳細を表示する ElastiCache CLI の例については、「」を参照してください [レプリケーショングループの詳細の表示 \(AWS CLI\)](#)。

Redis (クラスターモードが有効) クラスター (ElastiCache API) の詳細の表示

Redis (クラスターモードが有効) レプリケーショングループの詳細を表示する ElastiCache API の例については、「」を参照してください[レプリケーショングループの詳細の表示 \(ElastiCache API\)](#)。

レプリケーショングループの詳細の表示 (AWS CLI)

AWS CLI `describe-replication-groups` コマンドを使用してレプリケーショングループの詳細を表示できます。一覧を絞り込むには、以下のオプションパラメータを使用します。パラメータを省略すると、最大 100 個のレプリケーショングループの詳細が返されます。

オプションのパラメータ

- `--replication-group-id` – 特定のレプリケーショングループの詳細を表示するには、このパラメータを使用します。指定されたレプリケーショングループに複数のノードグループがある場合、結果はノードグループ別にグループ分けされて返されます。
- `--max-items` – 表示されるレプリケーショングループの数を制限するには、このパラメータを使用します。`--max-items` の値は 20 未満、または 100 を超えることはできません。

Example

次のコードは、最大 100 個のレプリケーショングループの詳細を表示します。

```
aws elasticache describe-replication-groups
```

次のコードは `sample-repl-group` の詳細を一覧します。

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

次のコードは `sample-repl-group` の詳細を一覧します。

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

次のコードリストは、最大 25 個のレプリケーショングループを示します。

```
aws elasticache describe-replication-groups --max-items 25
```

このオペレーションからの出力は以下のような JSON 形式になります。

```
{
```



```
"ReplicationGroups": [
  {
    "Status": "available",
    "Description": "test",
    "NodeGroups": [
      {
        "Status": "available",
        "NodeGroupMembers": [
          {
            "CurrentRole": "primary",
            "PreferredAvailabilityZone": "us-west-2a",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "rg-name-001.1abc4d.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "rg-name-001"
          },
          {
            "CurrentRole": "replica",
            "PreferredAvailabilityZone": "us-west-2b",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "rg-name-002.1abc4d.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "rg-name-002"
          },
          {
            "CurrentRole": "replica",
            "PreferredAvailabilityZone": "us-west-2c",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "rg-name-003.1abc4d.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "rg-name-003"
          }
        ],
        "NodeGroupId": "0001",
        "PrimaryEndpoint": {
          "Port": 6379,
          "Address": "rg-name.1abc4d.ng.0001.usw2.cache.amazonaws.com"
        }
      }
    ]
  }
]
```

```
    }
  ],
  "ReplicationGroupId": "rg-name",
  "AutomaticFailover": "enabled",
  "SnapshottingClusterId": "rg-name-002",
  "MemberClusters": [
    "rg-name-001",
    "rg-name-002",
    "rg-name-003"
  ],
  "PendingModifiedValues": {}
},
{
  ... some output omitted for brevity
}
]
```

詳細については、「AWS CLI for ElastiCache トピック [describe-replication-groups](#)」を参照してください。

レプリケーショングループの詳細の表示 (ElastiCache API)

AWS CLI DescribeReplicationGroups オペレーションを使用してレプリケーションの詳細を表示できます。一覧を絞り込むには、以下のオプションパラメータを使用します。パラメータを省略すると、最大 100 個のレプリケーショングループの詳細が返されます。

オプションのパラメータ

- **ReplicationGroupId** – 特定のレプリケーショングループの詳細を表示するには、このパラメータを使用します。指定されたレプリケーショングループに複数のノードグループがある場合、結果はノードグループ別にグループ分けされて返されます。
- **MaxRecords** – 表示されるレプリケーショングループの数を制限するには、このパラメータを使用します。MaxRecords の値は 20 未満、または 100 を超えることはできません。デフォルトは 100 です。

Example

次のコードリストは、最大 100 個のレプリケーショングループを示します。

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DescribeReplicationGroups
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

次のコードは myReplGroup の詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReplicationGroups
&ReplicationGroupId=myReplGroup
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReplicationGroups
&MaxRecords=25
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

詳細については、ElastiCache API リファレンストピック「[DescribeReplicationGroups](#)」を参照してください。

レプリケーショングループのエンドポイントの検索

アプリケーションは、ノードの DNS エンドポイントとポート番号がある場合、レプリケーショングループ内の任意のノードに接続できます。Redis (クラスターモードが無効) レプリケーショングループを実行しているか、Redis (クラスターモードが有効) レプリケーショングループを実行しているかにより、関心のあるエンドポイントが異なります。

Redis (クラスターモードが無効)

レプリカがある Redis (クラスターモードが無効) クラスターには、プライマリエンドポイント、読み込みエンドポイント、ノードエンドポイントの 3 種類のエンドポイントがあります。プライマリエンドポイントは、常にクラスターのプライマリノードに解決される DNS 名です。プライマリエンドポイントは、リードレプリカのプライマリロールへの昇格など、クラスターに対する変更の影響を受けません。書き込みアクティビティの場合、アプリケーションをプライマリエンドポイントに接続することをお勧めします。

リーダーエンドポイントは、エンドポイントへの受信接続を for ElastiCache Redis クラスター内のすべてのリードレプリカ間で均等に分割します。アプリケーションがいつ接続を作成するか、アプリケーションが接続をどのように (再) 利用するかなどの追加要因によって、トラフィックの分散が決定されます。レプリカが追加または削除されても、読み込みエンドポイントはリアルタイムでクラスターの変更に対応します。ElastiCache for Redis クラスターの複数のリードレプリカを異なるアベイラ AWS ビリティーゾーン (AZ) に配置して、リーダーエンドポイントの高可用性を確保できます。

Note

リーダーエンドポイントはロードバランサーではありません。これは、ラウンドロビン方式でレプリカノードの 1 つの IP アドレスに解決される DNS レコードです。

読み取りアクティビティの場合、アプリケーションはクラスター内のいずれのノードにも接続できます。プライマリエンドポイントとは異なり、ノードエンドポイントは特定のエンドポイントに解決されます。レプリカの追加または削除など、クラスターに変更を加えた場合は、アプリケーションでノードエンドポイントを更新する必要があります。

Redis (クラスターモードが有効)

レプリカがある Redis (クラスターモードが有効) クラスターには、複数のシャード (API/CLI: ノードグループ) があり、これはプライマリノードが複数あることを意味するため、Redis (クラスター

モードが無効) クラスターとはエンドポイント構造が異なります。Redis (クラスターモードが有効) には、クラスター内のすべてのプライマリエンドポイントとノードエンドポイントを「知っている」[設定エンドポイント]があります。アプリケーションは設定エンドポイントに接続します。アプリケーションからクラスターの設定エンドポイントに書き込みまたは読み取りを行うたびに、Redis は背後で、キーが属するシャードと、そのシャードで使用するエンドポイントを決定します。これはすべてアプリケーションに対して透過的です。

クラスターのエンドポイントは、ElastiCache コンソール、AWS CLI または ElastiCache API を使用して検索できます。

レプリケーショングループのエンドポイントの検索

レプリケーショングループのエンドポイントを確認するには、以下のトピックのいずれかを参照してください。

- [Redis \(クラスターモードが無効\) クラスターのエンドポイント \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) クラスターのエンドポイントを検索する \(コンソール\)](#)
- [レプリケーショングループのエンドポイントの検索 \(AWS CLI\)](#)
- [レプリケーショングループのエンドポイントの検索 \(ElastiCache API\)](#)

レプリケーショングループの変更

⚠ 重要な制約

- 現在、は、API オペレーション (CLI:) を使用したエンジンバージョンの変更など、Redis ModifyReplicationGroup (クラスターモードが有効) レプリケーショングループの限定的な変更 ElastiCache をサポートしています modify-replication-group。API オペレーション [ModifyReplicationGroupShardConfiguration](#) (CLI: [modify-replication-group-shard-configuration](#)) を用いて、Redis (クラスターモードが有効) クラスター内のシャード (ノードグループ) の数を変更できます。詳細については、「[Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)」を参照してください。

Redis (クラスターモードが有効) クラスターに他の変更を加える必要がある場合は、変更を組み込んだ新しいクラスターを作成します。

- Redis (クラスターモードが無効) と Redis (クラスターモードが有効) のクラスターとレプリケーショングループを新しいエンジンバージョンにアップグレードできます。ただし、既存のクラスターまたはレプリケーショングループを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。詳細については、「[エンジンバージョンとアップグレード](#)」を参照してください。
- 以下の例に示すように、コンソール、[ModifyReplicationグループ](#) API、または [modify-replication-group](#) CLI コマンドを使用して、クラスターモードが無効になっている既存の ElastiCache for Redis クラスターをアップグレードして、クラスターモードを有効にできます。または、「[クラスターモードの変更](#)」の手順に従うこともできます。

Redis (クラスターモードが無効) クラスターの設定は AWS CLI、ElastiCache コンソール、または ElastiCache API を使用して変更できます。現在、は、Redis (クラスターモードが有効) レプリケーショングループで限られた数の変更 ElastiCache をサポートしています。その他の変更では、現在のレプリケーショングループのバックアップを作成し、そのバックアップを使用して新しい Redis (クラスターモードが有効) レプリケーショングループをシードする必要があります。

トピック

- [の使用 AWS Management Console](#)
- [の使用 AWS CLI](#)
- [ElastiCache API の使用](#)

の使用 AWS Management Console

Redis (クラスターモードが無効) クラスターを変更するには、「[ElastiCache クラスターの変更](#)」を参照してください。

の使用 AWS CLI

modify-replication-group コマンド AWS CLI の例を次に示します。レプリケーショングループのそのほかの変更も同じコマンドを使用できます。

既存の Redis レプリケーショングループでマルチ AZ を有効にする:

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id myReplGroup \  
  --multi-az-enabled = true
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id myReplGroup ^  
  --multi-az-enabled
```

クラスターモードを無効から有効に変更する:

クラスターモードを [無効] から [有効] に変更するには、まずクラスターモードを [互換] に設定する必要があります。互換モードでは、Redis クライアントはクラスターモード有効およびクラスターモード無効の両方を使用して接続することができます。すべての Redis クライアントを移行してクラスターモードを有効にしたら、クラスターモードの設定を完了し、クラスターモードを [有効] に設定できます。

Linux、macOS、Unix の場合:

クラスターモードを [互換] に設定します。

```
aws elasticache modify-replication-group \  
  --replication-group-id myReplGroup \  
  --cache-parameter-group-name myParameterGroupName \  
  --cluster-mode compatible
```

クラスターモードを [有効] に設定します。

```
aws elasticache modify-replication-group \  
  --replication-group-id myReplGroup \  
  --cluster-mode enabled
```

Windows の場合:

クラスターモードを [互換] に設定します。

```
aws elasticache modify-replication-group ^  
  --replication-group-id myReplGroup ^  
  --cache-parameter-group-name myParameterGroupName ^  
  --cluster-mode compatible
```

クラスターモードを [有効] に設定します。

```
aws elasticache modify-replication-group ^  
  --replication-group-id myReplGroup ^  
  --cluster-mode enabled
```

コマンドの詳細については、AWS CLI [modify-replication-group](#)「[modify-replication-group](#)」または「for ElastiCache Redis [ユーザーガイド](#)」の「[クラスターモードの変更](#)」を参照してください。

ElastiCache API の使用

次の ElastiCache API オペレーションは、既存の Redis レプリケーショングループでマルチ AZ を有効にします。同じオペレーションを使用して、レプリケーショングループに対する他の変更を行うこともできます。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyReplicationGroup  
&AutomaticFailoverEnabled=true  
&Multi-AZEnabled=true  
&ReplicationGroupId=myReplGroup  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20141201T220302Z  
&Version=2014-12-01  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
```



```
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

ElastiCache API ModifyReplicationGroupオペレーションの詳細については、「」を参照してください [ModifyReplicationGroup](#)。

レプリケーショングループの削除

レプリカを持つクラスター (API/CLI ではレプリケーショングループ) のいずれかが不要になった場合は、それを削除できます。レプリケーショングループを削除すると、ElastiCache によってそのグループ内のすべてのノードが削除されます。

このオペレーションを開始した後は、中断またはキャンセルすることはできません。

Warning

ElastiCache for Redis クラスターを削除しても、手動スナップショットは保持されます。クラスターが削除される前に最終スナップショットを作成するオプションもあります。自動キャッシュスナップショットは保持されません。

レプリケーショングループの削除 (コンソール)

レプリカがあるクラスターを削除するには、「[クラスターの削除](#)」を参照してください。

レプリケーショングループの削除 (AWS CLI)

レプリケーショングループを削除するには、[delete-replication-group](#) コマンドを使用します。

```
aws elasticache delete-replication-group --replication-group-id my-repgroup
```

決定を確認するメッセージが表示されます。すぐにオペレーションを開始する場合は「y」(Yes) と入力します。プロセスの開始後に元に戻すことはできません。

```
After you begin deleting this replication group, all of its nodes will be deleted as well.
```

```
Are you sure you want to delete this replication group? [Ny]y
```

```
REPLICATIONGROUP my-repgroup My replication group deleting
```

レプリケーショングループの削除 (ElastiCache API)

[DeleteReplicationGroup](#) パラメータを使って `ReplicationGroup` を呼び出します。

Example

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DeleteReplicationGroup
&ReplicationGroupId=my-repgroup
&Version=2014-12-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Note

`RetainPrimaryCluster` パラメータを `true` に設定した場合、リードレプリカはすべて削除されますが、プライマリクラスターは保持されます。

レプリカの数の変更

、 、 AWS CLI または ElastiCache API を使用して AWS Management Console、Redis レプリケーショングループのリードレプリカの数を実動的に増減できます。レプリケーショングループが Redis (クラスターモードが有効) レプリケーショングループの場合、どのシャード (ノードグループ) のレプリカの数を増減するかを選択できます。

Redis レプリケーショングループのレプリカの数を実動的に変更するには、以下の状況に適したテーブルからオペレーションを選択します。

目的	Redis (クラスターモードが有効) の場合	Redis (クラスターモードが無効) の場合
レプリカの追加	シャードのレプリカの数を増やす	シャードのレプリカの数を増やす Redis (クラスターモードが無効) レプリケーショングループのリードレプリカの追加
レプリカの削除	シャードのレプリカの数を減らす	シャードのレプリカの数を減らす Redis (クラスターモードが無効) レプリケーショングループのリードレプリカの削除

シャードのレプリカを増やす

Redis (クラスターモードが有効) シャード、または Redis (クラスターモード無効) レプリケーショングループのレプリカ数を最大 5 個まで増やすことができます。これを行うには、AWS Management Console、AWS CLI、または ElastiCache API を使用します。

トピック

- [AWS Management Console を使用する場合](#)
- [AWS CLI を使用する場合](#)
- [ElastiCache API の使用](#)

AWS Management Console を使用する場合

次の手順では、Redis (クラスターモードが有効) レプリケーショングループのレプリカを増やすためにコンソールを使用します。

Redis シャードのレプリカを増やすには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis] を選択し、レプリカを追加するレプリケーショングループの名前を選択します。
3. レプリカを追加する各シャードのチェックボックスをオンにします
4. [Add replicas (レプリカの追加)] を選択します。
5. [Add Replicas to Shards (シャードにレプリカを追加)] ページを完了します。
 - [New number of replicas/shard (シャード当たりの新しいレプリカ数)] に、選択したすべてのシャードが持つようになるレプリカの数を入力します。この値は、[Current Number of Replicas per shard (現在のシャード当たりのレプリカ数)] 以上で、5 以下である必要があります。最小の実行として、レプリカを少なくとも 2 つにすることを勧めます。
 - [アベイラビリティゾーン] では、[指定なし] を選択して新しいレプリカごとにアベイラビリティゾーンを ElastiCache が選択するようにするか、アベイラビリティゾーンの指定] を選択して、新しいレプリカごとにアベイラビリティゾーンを選択します。

[Specify Availability Zones (アベイラビリティゾーンを指定する)] を選択した場合、リストを使用して新しい各レプリカのアベイラビリティゾーンを指定します。

6. [Add (追加)] を選択してレプリカを追加するか、[Cancel (キャンセル)] を選択してオペレーションをキャンセルします。

AWS CLI を使用する場合

Redis シャードでレプリカを増やすには、以下のパラメータを設定して `increase-replica-count` コマンドを使用します。

- `--replication-group-id` - 必須。レプリカを増やすレプリケーショングループを指定します。
- `--apply-immediately` または `--no-apply-immediately` - 必須。レプリカの数すぐに増やすか (`--apply-immediately`)、次のメンテナンスウィンドウで増やすか (`--no-apply-immediately`) を指定します。現在、`--no-apply-immediately` はサポートされていません。
- `--new-replica-count` - オプション。完了時のレプリカノードの数を、最大 5 個で指定します。ノードグループが 1 つだけある Redis (クラスターモードが無効) レプリケーショングループか Redis (クラスターモードが有効) グループで、または、レプリカの数と同じにするすべてのノードグループで、このパラメータを使用します。この値がノードグループの現在のレプリカの数より大きくない場合、呼び出しは失敗し、例外が発生します。
- `--replica-configuration` - オプション。各ノードグループのレプリカの数およびアベイラビリティゾーンを個別に設定できます。このパラメータを、各ノードグループを個別に設定する Redis (クラスターモードが有効) グループに使用します。

`--replica-configuration` には 3 つのオプションのメンバーがあります。

- `NodeId` - 設定するノードグループの 4 桁の ID。Redis (クラスターモードが無効) レプリケーショングループでは、シャード ID は常に 0001 です。Redis (クラスターモードが有効) ノードグループの (シャード) の ID を見つけるには、「[シャードの ID を見つける](#)」を参照してください。
- `NewReplicaCount` - このノードグループの、このオペレーションの最後のレプリカの数。この値は、最大 5 で、現在のレプリカの数より大きくなければなりません。この値がノードグループの現在のレプリカの数より大きくない場合、呼び出しは失敗し、例外が発生します。
- `PreferredAvailabilityZones` - レプリケーショングループのノードが存在するアベイラビリティゾーンを指定する `PreferredAvailabilityZone` 文字列のリスト。`PreferredAvailabilityZone` 値の数は、プライマリノードに対応する `NewReplicaCount` に 1 を足した数と等しい必要があります。`--replica-configuration` のこのメンバーを省略すると、新しいレプリカごとに、ElastiCache for Redis がアベイラビリティゾーンを選択します。

⚠ Important

呼び出しに、`--new-replica-count` または `--replica-configuration` パラメータのいずれかを含める必要がありますが、両方を含めることはできません。

Example

次の例では、`sample-repl-group` レプリケーショングループでレプリカの数 を 3 個に増やします。この例が終了すると、各ノードグループのレプリカは 3 個になります。この数は、単一のノードグループがある Redis (クラスターモードが無効) グループ、または、複数のノードグループがある Redis (クラスターモードが有効) グループのどちらにも適用されます。

Linux、macOS、Unix の場合:

```
aws elasticache increase-replica-count \  
  --replication-group-id sample-repl-group \  
  --new-replica-count 3 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache increase-replica-count ^  
  --replication-group-id sample-repl-group ^  
  --new-replica-count 3 ^  
  --apply-immediately
```

次の例では、`sample-repl-group` レプリケーショングループで、レプリカの数 を、2 つの指定されたノードグループの指定値に増やします。複数のノードグループがある場合、これは Redis (クラスターモードが有効) レプリケーショングループです。オプションの `PreferredAvailabilityZones` を指定する場合、リストされているアベイラビリティゾーンの数 は、`NewReplicaCount` に 1 以上を足した値と等しい必要があります。このアプローチは、`NodeGroupId` で指定されるグループのプライマリノードに対応します。

Linux、macOS、Unix の場合:

```
aws elasticache increase-replica-count \  
  --replication-group-id sample-repl-group \  
  --replica-configuration \  
  --new-replica-count 3
```

```
NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c,us-east-1b \  
NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c \  
--apply-immediately
```

Windows の場合:

```
aws elasticache increase-replica-count ^  
--replication-group-id sample-repl-group ^  
--replica-configuration ^  
NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c,us-east-1b ^  
NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c \  
--apply-immediately
```

CLI を使用したレプリカの数を増やす詳細については、Amazon ElastiCache コマンドラインリファレンスの「[increase-replica-count](#)」を参照してください。

ElastiCache API の使用

Redis シャードでレプリカの数を増やすには、以下のパラメータを設定して IncreaseReplicaCount アクションを使用します。

- ReplicationGroupId – 必須。レプリカの数を増やすレプリケーショングループを指定します。
- ApplyImmediately – 必須。レプリカの数をもとに増やすか (ApplyImmediately=True)、次のメンテナンスウィンドウで増やすか (ApplyImmediately=False) を指定します。現在、ApplyImmediately=False はサポートされていません。
- NewReplicaCount - オプション。完了時のレプリカノードの数を、最大 5 個で指定します。ノードグループが 1 つだけある Redis (クラスターモードが無効) レプリケーショングループで、または、すべてのノードグループでレプリカの数と同じにする Redis (クラスターモードが有効) グループで、このパラメータを使用します。この値がノードグループの現在のレプリカの数より大きくない場合、呼び出しは失敗し、例外が発生します。
- ReplicaConfiguration - オプション。各ノードグループのレプリカの数およびアベイラビリティゾーンを個別に設定できます。このパラメータを、各ノードグループを個別に設定する Redis (クラスターモードが有効) グループに使用します。

ReplicaConfiguraion には 3 つのオプションのメンバーがあります。

- **NodeId** – 設定するノードグループの 4 桁の ID。Redis (クラスターモードが無効) レプリケーショングループでは、ノードグループ (シャード) ID は常に 0001 です。Redis (クラスターモードが有効) ノードグループの (シャード) の ID を見つけるには、「[シャードの ID を見つける](#)」を参照してください。
- **NewReplicaCount** – このノードグループの、このオペレーションの最後のレプリカの数。この値は、最大 5 で、現在のレプリカの数より大きくなければなりません。この値がノードグループの現在のレプリカの数より大きくない場合、呼び出しは失敗し、例外が発生します。
- **PreferredAvailabilityZones** – レプリケーショングループのノードが存在するアベイラビリティゾーンを指定する PreferredAvailabilityZone 文字列のリスト。PreferredAvailabilityZone 値の数は、プライマリノードに対応する NewReplicaCount に 1 を足した数と等しい必要があります。ReplicaConfiguration のこのメンバーを省略すると、新しいレプリカごとに、ElastiCache for Redis がアベイラビリティゾーンを選択します。

Important

呼び出しに、NewReplicaCount または ReplicaConfiguration パラメータのいずれかを含める必要がありますが、両方を含めることはできません。

Example

次の例では、sample-repl-group レプリケーショングループでレプリカ数を 3 個に増やします。この例が終了すると、各ノードグループのレプリカは 3 個になります。この数は、単一のノードグループがある Redis (クラスターモードが無効) グループ、または、複数のノードグループがある Redis (クラスターモードが有効) グループのどちらにも適用されます。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=IncreaseReplicaCount  
  &ApplyImmediately=True  
  &NewReplicaCount=3  
  &ReplicationGroupId=sample-repl-group  
  &Version=2015-02-02  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20150202T192317Z  
  &X-Amz-Credential=<credential>
```

次の例では、sample-repl-group レプリケーショングループで、レプリカの数、2 つの指定されたノードグループの指定値を増やします。複数のノードグループがある場合、これは Redis (クラスターモードが有効) レプリケーショングループです。オプションの PreferredAvailabilityZones を指定する場合、リストされているアベイラビリティゾーンの数、NewReplicaCount に 1 以上を足した値と等しい必要があります。このアプローチは、NodeGroupId で指定されるグループのプライマリノードに対応します。

```
https://elasticache.us-west-2.amazonaws.com/
  ?Action=IncreaseReplicaCount
  &ApplyImmediately=True
  &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
  &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=2

  &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

  &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c

  &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1b
  &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
  &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=3

  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1c

  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
  &ReplicationGroupId=sample-repl-group
  &Version=2015-02-02
  &SignatureVersion=4
  &SignatureMethod=HmacSHA256
  &Timestamp=20150202T192317Z
  &X-Amz-Credential=<credential>
```

API を使用したレプリカの数を増やす詳細については、Amazon ElastiCache API リファレンスの「[IncreaseReplicaCount](#)」を参照してください。

シャードのレプリカの数減らす

Redis (クラスターモードが有効) のシャード、または Redis (クラスターモードが無効) のレプリケーショングループのレプリカ数を減らすことができます。

- Redis (クラスターモードが無効) のレプリカ数では、マルチ AZ が有効な場合に 1 まで減らし、マルチ AZ が有効でない場合にゼロまで減らすことができます。
- Redis (クラスターモードが有効) では、レプリカ数をゼロまで減らすことができます。ただし、プライマリノードが失敗した場合はレプリカにフェイルオーバーできません。

AWS Management Console、AWS CLI または ElastiCache API を使用して、ノードグループ (シャード) またはレプリケーショングループのレプリカ数を減らすことができます。

トピック

- [の使用 AWS Management Console](#)
- [の使用 AWS CLI](#)
- [ElastiCache API の使用](#)

の使用 AWS Management Console

次の手順では、Redis (クラスターモードが有効) レプリケーショングループのレプリカ数を減らすためにコンソールを使用します。

Redis シャードのレプリカ数を減らすには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis] を選択し、レプリカを減らすレプリケーショングループの名前を選択します。
3. レプリカノードを削除する各シャードのチェックボックスをオンにします
4. [Delete replicas (レプリカの削除)] を選択します。
5. [Delete Replicas from Shards (シャードからのレプリカの削除)] ページを完了します。
 - a. [New number of replicas/shard (シャード当たりの新しいレプリカ数)] に、選択したシャードが持つようになるレプリカ数を入力します。この数は、1 以上にする必要があります。最小の実行として、レプリカを少なくともシャードあたり 2 つにすることをお勧めします。

- b. [Delete (削除)] を選択してレプリカを削除するか、[Cancel (キャンセル)] を選択してオペレーションをキャンセルします。

⚠ Important

- 削除するレプリカノードを指定しない場合、Redis ElastiCache では削除対象のレプリカノードが自動的に選択されます。その間、ElastiCache Redis はレプリケーショングループのマルチ AZ アーキテクチャを保持し、その後にプライマリでレプリケーションラグを最小限に抑えてレプリカを保持しようとしています。
- プライマリまたはレプリケーショングループ内のプライマリノードを削除することはできません。プライマリノードを削除するように指定すると、オペレーションは失敗し、プライマリノードが削除のために選択されたことを示すエラーイベントが示されます。

の使用 AWS CLI

Redis シャードでレプリカの数減らすには、以下のパラメータを設定して `decrease-replica-count` コマンドを使用します。

- `--replication-group-id` - 必須。レプリカの数減らすレプリケーショングループを指定します。
- `--apply-immediately` または `--no-apply-immediately` - 必須。レプリカの数減らすか (`--apply-immediately`)、次のメンテナンスウィンドウで減らすか (`--no-apply-immediately`) を指定します。現在、`--no-apply-immediately` はサポートされていません。
- `--new-replica-count` - オプション。レプリカノードの数を指定します。`--new-replica-count` の値は、ノードグループの現在のレプリカの数よりも小さい有効な値である必要があります。最小許容値については、「[シャードのレプリカの数減らす](#)」を参照してください。`--new-replica-count` の値がこの要件を満たさない場合、呼び出しは失敗します。
- `--replicas-to-remove` - オプション。削除するレプリカノードを指定するノード ID のリストが含まれます。
- `--replica-configuration` - オプション。各ノードグループのレプリカの数およびアベイラビリティゾーンを個別に設定できます。このパラメータを、各ノードグループを個別に設定する Redis (クラスターモードが有効) グループに使用します。

`--replica-configuration` には 3 つのオプションのメンバーがあります。

- `NodeGroupId` – 設定するノードグループの 4 桁の ID。Redis (クラスターモードが無効) レプリケーショングループでは、シャード ID は常に 0001 です。Redis (クラスターモードが有効) ノードグループの (シャード) の ID を見つけるには、「[シャードの ID を見つける](#)」を参照してください。
- `NewReplicaCount` – レプリカノードの数を指定するオプションのパラメータ。`NewReplicaCount` の値は、ノードグループの現在のレプリカの数よりも小さい有効な値である必要があります。最小許容値については、「[シャードのレプリカの数減らす](#)」を参照してください。`NewReplicaCount` の値がこの要件を満たさない場合、呼び出しは失敗します。
- `PreferredAvailabilityZones` – レプリケーショングループのノードが存在するアベイラビリティゾーンを指定する `PreferredAvailabilityZone` 文字列のリスト。`PreferredAvailabilityZone` 値の数は、プライマリノードに対応する `NewReplicaCount` に 1 を足した数と等しい必要があります。このメンバー `--replica-configuration` を省略すると、Redis ElastiCache は新しいレプリカごとにアベイラビリティゾーンを選択します。

Important

`--new-replica-count`、`--replicas-to-remove`、または `--replica-configuration` パラメータのいずれか 1 つのみを含める必要があります。

Example

次の例では、`--new-replica-count` を使用して、`sample-repl-group` レプリケーショングループのレプリカ数を 1 個に減らします。この例が終了すると、各ノードグループのレプリカは 1 個になります。この数は、単一のノードグループがある Redis (クラスターモードが無効) グループ、または、複数のノードグループがある Redis (クラスターモードが有効) グループのどちらにも適用されます。

Linux、macOS、Unix の場合:

```
aws elasticache decrease-replica-count
  --replication-group-id sample-repl-group \
  --new-replica-count 1 \
  --apply-immediately
```

Windows の場合:

```
aws elasticache decrease-replica-count ^
  --replication-group-id sample-repl-group ^
  --new-replica-count 1 ^
  --apply-immediately
```

次の例では、ノードグループから 2 つの指定されたレプリカ (0001 と 0003) を削除して、`sample-repl-group` レプリケーショングループのレプリカの数減らします。

Linux、macOS、Unix の場合:

```
aws elasticache decrease-replica-count \
  --replication-group-id sample-repl-group \
  --replicas-to-remove 0001,0003 \
  --apply-immediately
```

Windows の場合:

```
aws elasticache decrease-replica-count ^
  --replication-group-id sample-repl-group ^
  --replicas-to-remove 0001,0003 \
  --apply-immediately
```

次の例では、`--replica-configuration` を使用して、`sample-repl-group` レプリケーショングループで、レプリカの数、2 つの指定されたノードグループの指定値に減らします。複数のノードグループがある場合、これは Redis (クラスターモードが有効) レプリケーショングループです。オプションの `PreferredAvailabilityZones` を指定する場合、リストされているアベイラビリティゾーンの数、`NewReplicaCount` に 1 以上を足した値と等しい必要があります。このアプローチは、`NodeGroupId` で指定されるグループのプライマリノードに対応します。

Linux、macOS、Unix の場合:

```
aws elasticache decrease-replica-count \
  --replication-group-id sample-repl-group \
  --replica-configuration \
    NodeGroupId=0001,NewReplicaCount=1,PreferredAvailabilityZones=us-east-1a,us-east-1c \
    NodeGroupId=0003,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c \
  --apply-immediately
```

Windows の場合:

```
aws elasticache decrease-replica-count ^
  --replication-group-id sample-repl-group ^
  --replica-configuration ^
    NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c ^
    NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c \
  --apply-immediately
```

CLI を使用してレプリカの数減らす方法の詳細については、「Amazon コマンドラインリファレンス」の「[decrease-replica-count](#)」を参照してください。ElastiCache

ElastiCache API の使用

Redis シャードでレプリカの数減らすには、以下のパラメータを設定して DecreaseReplicaCount アクションを使用します。

- ReplicationGroupId - 必須。レプリカの数減らすレプリケーショングループを指定します。
- ApplyImmediately - 必須。レプリカの数減らすか (ApplyImmediately=True)、次のメンテナンスウィンドウで減らすか (ApplyImmediately=False) を指定します。現在、ApplyImmediately=False はサポートされていません。
- NewReplicaCount - オプション。レプリカノードの数を指定します。NewReplicaCount の値は、ノードグループの現在のレプリカの数よりも小さい有効な値である必要があります。最小許容値については、「[シャードのレプリカの数減らす](#)」を参照してください。--new-replica-count の値がこの要件を満たさない場合、呼び出しは失敗します。
- ReplicasToRemove - オプション。削除するレプリカノードを指定するノード ID のリストが含まれます。
- ReplicaConfiguration - オプション。各ノードグループのレプリカの数およびアベイラビリティゾーンを個別に設定するノードグループのリストが含まれます。このパラメータを、各ノードグループを個別に設定する Redis (クラスターモードが有効) グループに使用します。

ReplicaConfiguraion には 3 つのオプションのメンバーがあります。

- NodeGroupId - 設定するノードグループの 4 桁の ID。Redis (クラスターモードが無効) レプリケーショングループでは、ノードグループ ID は常に 0001 です。Redis (クラスターモードが有効) ノードグループの (シャード) の ID を見つけるには、「[シャードの ID を見つける](#)」を参照してください。

- `NewReplicaCount` – このノードグループの、このオペレーションの最後のレプリカの数。この値は、現在のレプリカの数よりも小さくし、マルチ AZ が有効な場合は最小 1、自動フェイルオーバーを備えたマルチ AZ が有効でない場合は最小 0 にする必要があります。この値がノードグループの現在のレプリカの数より小さくない場合、呼び出しは失敗し、例外が発生します。
- `PreferredAvailabilityZones` – レプリケーショングループのノードが存在するアベイラビリティゾーンを指定する `PreferredAvailabilityZone` 文字列のリスト。`PreferredAvailabilityZone` 値の数は、プライマリノードに対応する `NewReplicaCount` に 1 を足した数と等しい必要があります。このメンバー `ReplicaConfiguration` を省略すると、Redis ElastiCache は新しいレプリカごとにアベイラビリティゾーンを選択します。

Important

`NewReplicaCount`、`ReplicasToRemove`、または `ReplicaConfiguration` パラメータのいずれか 1 つのみを含める必要があります。

Example

次の例では、`NewReplicaCount` を使用して、`sample-repl-group` レプリケーショングループのレプリカ数を 1 個に減らします。この例が終了すると、各ノードグループのレプリカは 1 個になります。この数は、単一のノードグループがある Redis (クラスターモードが無効) グループ、または、複数のノードグループがある Redis (クラスターモードが有効) グループのどちらにも適用されます。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=DecreaseReplicaCount  
  &ApplyImmediately=True  
  &NewReplicaCount=1  
  &ReplicationGroupId=sample-repl-group  
  &Version=2015-02-02  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20150202T192317Z  
  &X-Amz-Credential=<credential>
```

次の例では、ノードグループから 2 つの指定されたレプリカ (0001 と 0003) を削除して、`sample-repl-group` レプリケーショングループのレプリカ数を減らします。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=DecreaseReplicaCount  
  &ApplyImmediately=True  
  &ReplicasToRemove.ReplicaToRemove.1=0001  
  &ReplicasToRemove.ReplicaToRemove.2=0003  
  &ReplicationGroupId=sample-repl-group  
  &Version=2015-02-02  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20150202T192317Z  
  &X-Amz-Credential=<credential>
```

次の例では、ReplicaConfiguration を使用して、sample-repl-group レプリケーショングループで、レプリカの本数を、2 つの指定されたノードグループの指定値に減らします。複数のノードグループがある場合、これは Redis (クラスターモードが有効) レプリケーショングループです。オプションの PreferredAvailabilityZones を指定する場合、リストされているアベイラビリティゾーンの本数は、NewReplicaCount に 1 以上を足した値と等しい必要があります。このアプローチは、NodeGroupId で指定されるグループのプライマリノードに対応します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=DecreaseReplicaCount  
  &ApplyImmediately=True  
  &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001  
  &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=1  
  
  &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=  
east-1a  
  
  &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=  
east-1c  
  &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003  
  &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=2  
  
  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=  
east-1a  
  
  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=  
east-1b  
  
  &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=  
east-1c  
  &ReplicationGroupId=sample-repl-group
```

```
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

API を使用してレプリカの数減らす方法の詳細については、「Amazon API リファレンス」の [DecreaseReplica 「カウント」](#) を参照してください。 ElastiCache

Redis (クラスターモードが無効) レプリケーショングループのリードレプリカの追加

以下のトピックの情報は、Redis (クラスターモードが無効) レプリケーショングループにのみ適用されます。

読み取りトラフィックが増えるにつれて、これらの読み取りをより多くのノードに分散させて、1つのノードの読み取りの負荷を減らすことを考えます。このトピックでは、Redis (クラスターモードが無効) クラスターにリードレプリカを追加する方法について説明します。

Redis (クラスターモードが無効) レプリケーショングループは、最大 5 つのリードレプリカを持つことができます。すでに 5 個のリードレプリカを持つレプリケーショングループに別のリードレプリカを追加しようとする、オペレーションが失敗します。

Redis (クラスターモードが有効) レプリケーショングループへのレプリカの追加については、以下を参照してください。

- [Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)
- [シャードのレプリカの数を増やす](#)

Redis (クラスターモードが無効) クラスターにリードレプリカを追加するには、ElastiCache コンソール、AWS CLI、または ElastiCache API を使用します。

関連トピック

- [クラスターへのノードの追加](#)
- [レプリケーショングループへのリードレプリカの追加 \(AWS CLI\)](#)
- [API を使用したレプリケーショングループへのリードレプリカの追加](#)

レプリケーショングループへのリードレプリカの追加 (AWS CLI)

Redis (クラスターモードが無効) レプリケーショングループにリードレプリカを追加するには、AWS CLI `create-cache-cluster` コマンドを使用します。パラメータとして `--replication-group-id` を使用し、クラスター (ノード) を追加するレプリケーショングループを指定します。

次の例では、クラスター `my-read-replica` を作成して、レプリケーショングループ `my-replication-group` に追加します。リードレプリカのノードタイプ、パラメータグループ、セキュリティグループ、メンテナンスの時間などの設定は、`my-replication-group` の他のノードの設定と同じです。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-cluster \  
  --cache-cluster-id my-read-replica \  
  --replication-group-id my-replication-group
```

Windows の場合:

```
aws elasticache create-cache-cluster ^  
  --cache-cluster-id my-read-replica ^  
  --replication-group-id my-replication-group
```

CLI を使用したリードレプリカの追加の詳細については、Amazon ElastiCache コマンドラインリファレンスの「[create-cache-cluster](#)」を参照してください。

API を使用したレプリケーショングループへのリードレプリカの追加

Redis (クラスターモードが無効) レプリケーショングループにリードレプリカを追加するには、ElastiCache `CreateCacheCluster` オペレーションを使用します。パラメータとして `ReplicationGroupId` を使用し、クラスター (ノード) を追加するレプリケーショングループを指定します。

次の例では、クラスター `myReadReplica` を作成して、レプリケーショングループ `myReplicationGroup` に追加します。リードレプリカのノードタイプ、パラメータグループ、セキュリティグループ、メンテナンスの時間などの設定は、`myReplicationGroup` の他のノードの設定と同じです。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=CreateCacheCluster
```

```
&CacheClusterId=myReadReplica
&ReplicationGroupId=myReplicationGroup
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

API を使用したリードレプリカの追加の詳細については、Amazon ElastiCache API リファレンスの「[CreateCacheCluster](#)」を参照してください。

Redis (クラスターモードが無効) レプリケーショングループのリードレプリカの削除

以下のトピックの情報は、Redis (クラスターモードが無効) レプリケーショングループにのみ適用されます。

Redis レプリケーショングループの読み取りトラフィックを変更すると、リードレプリカの追加または削除を行うことができます。Redis (クラスターモードが無効) レプリケーショングループからノードを削除することには制限がありますが、単なるクラスターの削除と同じです。

- レプリケーショングループからプライマリを削除することはできません。プライマリを削除する場合は、以下を実行します。
 - リードレプリカをプライマリに昇格させます。リードレプリカをプライマリに昇格させる詳細については、「[Redis \(クラスターモード無効\) レプリケーショングループのリードレプリカのプライマリへの昇格](#)」を参照してください。
 - 古いプライマリを削除します。この方法の制限については、次のポイントを参照してください。
- レプリケーショングループでマルチ AZ が有効になっている場合は、そのレプリケーショングループから最後のリードレプリカを削除することはできません。この場合は次の操作を行います。
 - マルチ AZ を無効にしてレプリケーショングループを変更します。詳細については、「[レプリケーショングループの変更](#)」を参照してください。
 - リードレプリカを削除します。

Redis (クラスターモードが無効) レプリケーショングループからリードレプリカを削除するには、ElastiCache コンソール、AWS CLI for ElastiCache、またはElastiCache APIを使用します。

Redis レプリケーショングループからレプリカを削除する方法については、以下を参照してください。

- [AWS Management Console を使用する場合](#)
- [AWS CLIの使用](#)
- [ElastiCache API の使用](#)
- [Redis \(クラスターモードが有効\) でのクラスターのスケーリング](#)
- [シャードのレプリカの数減らす](#)

Redis (クラスターモード無効) レプリケーショングループのリードレプリカのプライマリへの昇格

以下のトピックの情報は、Redis (クラスターモードが無効) レプリケーショングループにのみ適用されます。

、 AWS CLIまたは ElastiCache API を使用して AWS Management Console、Redis (クラスターモードが無効) リードレプリカをプライマリに昇格させることができます。Redis (クラスターモード無効) レプリケーショングループでマルチ AZ と自動フェイルオーバーを有効にしている場合は、リードレプリカをプライマリに昇格させることはできません。マルチ AZ を有効にしているレプリケーショングループのプライマリに Redis (クラスターモード無効) レプリカを昇格させる場合は、以下が必要です。

1. レプリケーショングループを変更してマルチ AZ を無効にします (これを行うためにすべてのクラスターが同じアベイラビリティーゾーンに存在する必要はありません)。詳細については、「[レプリケーショングループの変更](#)」を参照してください。
2. リードレプリカをプライマリに昇格させます。
3. マルチ AZ を再び有効にするためにレプリケーショングループを変更します。

マルチ AZ は、Redis 2.6.13 以前を実行しているレプリケーショングループでは使用できません。

の使用 AWS Management Console

次の手順では、コンソールを使用してレプリカノードをプライマリに昇格させます。

リードレプリカをプライマリに昇格させるには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. 昇格させるレプリカが、マルチ AZ が有効な Redis (クラスターモード無効) レプリケーショングループのメンバーである場合は、先に進む前に、レプリケーショングループを変更してマルチ AZ を無効にします。詳細については、「[レプリケーショングループの変更](#)」を参照してください。
3. [Redis] を選択して、クラスターのリストから、変更するレプリケーショングループを選択します。このレプリケーショングループは、「Clustered Redis」エンジンではなく、「Redis」エンジンを実行していること、また 2 つ以上のノードを持っていることが必要です。
4. ノードのリストからプライマリに昇格させるレプリカノードを選択して、[Actions (アクション)] で、[Promote (昇格)] を選択します。

5. [Promote Read Replica (リードレプリカの昇格)] ダイアログボックスで、次の操作を行います。
 - a. [Apply Immediately (すぐに適用)] で、リードレプリカをすぐに昇格させる場合は [Yes (はい)] を選択し、クラスタの次のメンテナンス期間に昇格させる場合は [No (いいえ)] を選択します。
 - b. リードレプリカを昇格させる場合は [Promote] を選択し、オペレーションをキャンセルする場合は [No] を選択します。
6. 昇格プロセスを開始する前にクラスタでマルチ AZ を有効にしている場合は、レプリケーショングループのステータスが [available (利用可能)] になるまで待機して、マルチ AZ を再度有効に変更します。詳細については、「[レプリケーショングループの変更](#)」を参照してください。

の使用 AWS CLI

現在、レプリケーショングループでマルチ AZ を有効にしている場合はリードレプリカをプライマリに昇格させることはできません。場合によっては、昇格させるレプリカが、マルチ AZ が有効なレプリケーショングループのメンバーとなっていることがあります。この場合、続行する前に、レプリケーショングループを変更して、マルチ AZ を無効にする必要があります。これを行うためにすべてのクラスタが同じアベイラビリティーゾーンに存在する必要はありません。レプリケーショングループの変更の詳細については、「[レプリケーショングループの変更](#)」を参照してください。

次の AWS CLI コマンドは、レプリケーショングループ を変更し `sample-repl-group`、リードレプリカ `my-replica-1` をレプリケーショングループのプライマリにします。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id sample-repl-group \  
  --primary-cluster-id my-replica-1
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id sample-repl-group ^  
  --primary-cluster-id my-replica-1
```

レプリケーショングループの変更の詳細については、「Amazon コマンドラインリファレンス [modify-replication-group](#)」の「」を参照してください。 ElastiCache

ElastiCache API の使用

現在、レプリケーショングループでマルチ AZ を有効にしている場合はリードレプリカをプライマリに昇格させることはできません。場合によっては、昇格させるレプリカが、マルチ AZ が有効なレプリケーショングループのメンバーとなっていることがあります。この場合、続行する前に、レプリケーショングループを変更して、マルチ AZ を無効にする必要があります。これを行うためにすべてのクラスターが同じアベイラビリティーゾーンに存在する必要はありません。レプリケーショングループの変更の詳細については、「[レプリケーショングループの変更](#)」を参照してください。

次の ElastiCache API アクションは、レプリケーショングループを変更し myReplGroup、リードレプリカ myReplica-1 をレプリケーショングループのプライマリにします。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyReplicationGroup  
&ReplicationGroupId=myReplGroup  
&PrimaryClusterId=myReplica-1  
&Version=2014-12-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20141201T220302Z  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Date=20141201T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

レプリケーショングループの変更の詳細については、Amazon API リファレンス [ModifyReplicationGroup](#) の「」を参照してください。 ElastiCache

メンテナンスの管理

すべてのクラスターとレプリケーショングループには週次のメンテナンスウィンドウがあり、その期間内にシステムの変更が適用されます。クラスターまたはレプリケーショングループの作成または変更時に、希望するメンテナンスウィンドウを指定しない場合、ElastiCache では、ランダムに選択された曜日に対してリージョン内で 60 分のメンテナンスウィンドウを割り当てます。

60 分のメンテナンス時間は、リージョンごとに決められた 8 時間の中でランダムに選択されます。次の表に、デフォルトでメンテナンス時間が割り当てられる各リージョンの時間ブロックを示します。リージョンのメンテナンス時間外で、希望するメンテナンス時間を選択できます。

リージョンコード	リージョン名	リージョンメンテナンスウィンドウ
ap-northeast-1	アジアパシフィック (東京) リージョン	13:00 ~ 21:00 UTC
ap-northeast-2	アジアパシフィック (ソウル) リージョン	12:00 ~ 20:00 UTC
ap-northeast-3	アジアパシフィック (大阪) リージョン	12:00 ~ 20:00 UTC
ap-southeast-3	アジアパシフィック (ジャカルタ) リージョン	14:00 ~ 22:00 UTC
ap-south-1	アジアパシフィック (ムンバイ) リージョン	17:30 ~ 1:30 UTC
ap-southeast-1	アジアパシフィック (シンガポール) リージョン	14:00 ~ 22:00 UTC
cn-north-1	中国 (北京) リージョン	14:00 ~ 22:00 UTC
cn-northwest-1	中国 (寧夏) リージョン	14:00 ~ 22:00 UTC
ap-east-1	アジアパシフィック (香港) リージョン	13:00 ~ 21:00 UTC
ap-southeast-2	アジアパシフィック (シドニー) リージョン	12:00 ~ 20:00 UTC
eu-west-3	EU (パリ) リージョン	23:59 ~ 07:29 UTC
af-south-1	アフリカ (ケープタウン) リージョン	13:00 ~ 21:00 UTC
eu-central-1	欧州 (フランクフルト) リージョン	23:00 ~ 07:00 UTC
eu-west-1	欧州 (アイルランド) リージョン	22:00 ~ 06:00 UTC

リージョンコード	リージョン名	リージョンメンテナンスウィンドウ
eu-west-2	欧州 (ロンドン) リージョン	23:00 ~ 07:00 UTC
me-south-1	中東 (バーレーン) リージョン	13:00 ~ 21:00 UTC
me-central-1	中東 (アラブ首長国連邦) リージョン	13:00 ~ 21:00 UTC
eu-south-1	欧州 (ミラノ) リージョン	21:00 ~ 05:00 UTC
sa-east-1	南米 (サンパウロ) リージョン	01:00 ~ 09:00 UTC
us-east-1	米国東部 (バージニア北部) リージョン	03:00 ~ 11:00 UTC
us-east-2	米国東部 (オハイオ) リージョン	04:00 ~ 12:00 UTC
us-gov-west-1	AWS GovCloud (US) リージョン	06:00 ~ 14:00 UTC
us-west-1	US West (N. California) リージョン	06:00 ~ 14:00 UTC
us-west-2	米国西部 (オレゴン) リージョン	06:00 ~ 14:00 UTC

クラスターまたはレプリケーショングループのメンテナンスウィンドウの変更

メンテナンスウィンドウは使用率の最も低い時間帯に設定する必要があります。このため、場合によっては変更が必要になります。クラスターまたはレプリケーショングループを変更して、リクエストしたメンテナンス作業が発生するまでの時間範囲 (最大 24 時間) を指定することができます。お客様がリクエストした延期または保留中のクラスターの変更は、この時間に行われます。

Note

AWS Management Console を使用してノードタイプの変更やエンジンのアップグレードを直ちに適用する場合は、[Apply Immediately] (すぐに適用) ボックスを選択します。それ以外の場合は、次にスケジュールされているメンテナンス期間中にこれらの変更が適用されま

す。API を使用するには、「[modify-replication-group](#)」または「[modify-cache-cluster](#)」を参照してください。

詳細情報

メンテナンスウィンドウとノード交換の詳細については、以下を参照してください。

- [ElastiCache メンテナンス](#)—メンテナンスとノード交換のよくある質問
- [ノードの置換](#)—ノード交換の管理
- [レプリケーショングループの変更](#)—レプリケーショングループのメンテナンスウィンドウの変更

パラメータグループを使用したエンジンパラメータの設定

Amazon ElastiCache はパラメータを使用して、ノードとクラスターの実行時のプロパティを制御します。通常、新しいエンジンバージョンには新しい機能をサポートするための追加のパラメータが含まれます。パラメータのテーブルについては、「[Redis 固有のパラメータ](#)」を参照してください。

もちろん、maxmemory などのパラメータ値はエンジンやノードのタイプによって決まります。ノードタイプ別のパラメータ値のテーブルについては、「[Redis のノードタイプ固有のパラメータ](#)」を参照してください。

トピック

- [パラメータの管理](#)
- [キャッシュパラメータグループの階層](#)
- [パラメータグループを作成する](#)
- [パラメータグループを名前別に一覧表示する](#)
- [パラメータグループの値を一覧表示する](#)
- [パラメータグループを変更する](#)
- [パラメータグループを削除する](#)
- [Memcached 固有のパラメータ](#)
- [Redis 固有のパラメータ](#)

パラメータの管理

パラメータの管理を容易にするために、パラメータは名前付きのパラメータグループに分類されます。パラメータグループは、起動時にエンジンソフトウェアに渡されるパラメータの特定の値の組み合わせを表しています。これらの値により、各ノードのエンジンプロセスが実行時にどのように動作するかが決まります。特定のパラメータグループのパラメータ値は、クラスターが属するグループに関係なく、そのグループに関連付けられているすべてのノードに適用されます。

クラスターのパフォーマンスを最適化するには、パラメータ値を変更するか、またはクラスターのパラメータグループを変更できます。

- デフォルトのパラメータグループの変更や削除はできません。カスタムパラメータ値が必要な場合は、独自のパラメータグループを作成する必要があります。
- パラメータグループファミリーとユーザーが割り当てているクラスターには、互換性が必要です。たとえば、クラスターが Redis バージョン 3.2.10 を実行している場合は、Redis3.2 ファミリーのパラメータグループ、デフォルト値またはカスタム値のみを使用できます。
- クラスターのパラメータグループを変更する場合は、条件付きで変更可能なパラメータの値は、現在のパラメータグループと新しいパラメータグループで一致している必要があります。
- クラスターのパラメータを変更すると、変更はすぐにクラスターに適用されるか、次に挙げる例外を除き、クラスターノードの再起動後に適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。特定のパラメータの変更が適用されるタイミングを確認するには、[Redis 固有のパラメータ](#) のテーブルの「変更が有効になる」列を参照してください。

詳細については、「[ノードの再起動](#)」を参照してください。

Redis (クラスターモードが有効) パラメータの変更

Redis (クラスターモードが有効) クラスターで次のパラメータを変更する場合は、以降のステップに従います。

- アクティブハッシュ化
 - データベース
1. クラスターの手動バックアップを作成します。「[手動バックアップの取得](#)」を参照してください。
 2. Redis (クラスターモードが有効) クラスターを削除します。「[クラスターの削除](#)」を参照してください。

3. 変更されたパラメータグループとバックアップを使用してクラスターを復元し、新しいクラスターをシードします。「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

他のパラメータを変更する場合、これは必要ありません。

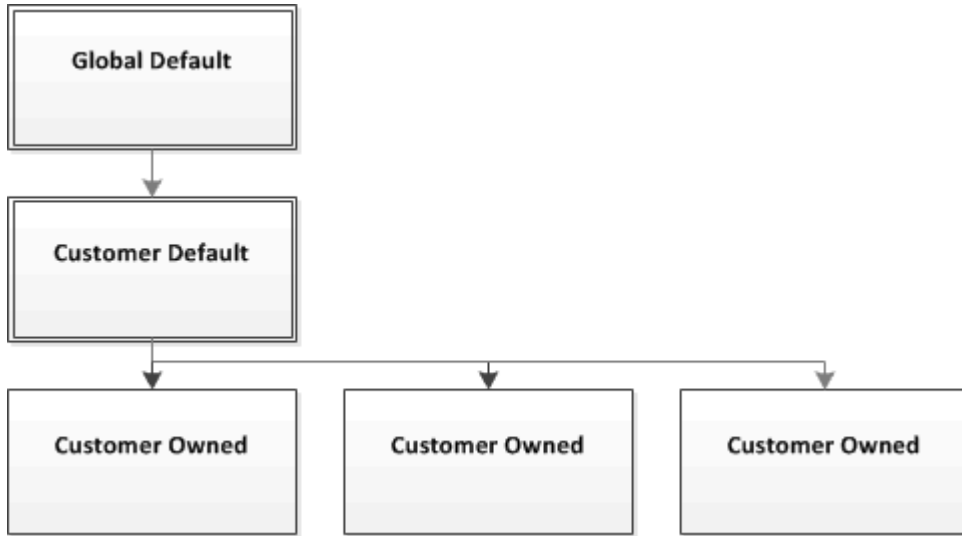
- パラメータグループを Redis Global Datastore に関連付けることができます。[Global Datastore] は、複数の AWS リージョンにまたがる 1 つ以上のクラスターのコレクションです。この場合、パラメータグループは、Global Datastore を構成するすべてのクラスターで共有されます。プライマリクラスターのパラメータグループに対する変更は、Global Datastore 内の残りのすべてのクラスターにレプリケートされます。詳細については、「[グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)」を参照してください。

パラメータグループが Global Datastore の一部であるかどうかを確認するには、次の場所を調べます。

- [パラメータグループ] ページの ElastiCache コンソールで、yes/no [グローバル] 属性
- [\[CacheParameterGroup\]](#) API オペレーションの yes/no IsGlobal プロパティの

キャッシュパラメータグループの階層

Amazon ElastiCache には、以下に示すキャッシュパラメータグループの 3 つの階層があります。



Amazon ElastiCache パラメータグループの階層

グローバルデフォルト

リージョン内のすべての Amazon ElastiCache のお客様向け最上位ルートパラメータグループ。

グローバルデフォルトのキャッシュパラメータグループ:

- ElastiCache 向けに確保されており、お客様が使用することはできません。

お客様デフォルト

グローバルデフォルトのキャッシュパラメータグループのコピーは、お客様が使用するために作成されています。

お客様デフォルトのキャッシュパラメータグループ:

- ElastiCache によって作成され、所有されています。
- このキャッシュパラメータグループでサポートされているエンジンのバージョンを実行しているすべてのクラスターのキャッシュパラメータグループとして使用できます。
- お客様が編集することはできません。

お客様所有

お客様デフォルトのキャッシュパラメータグループのコピー。お客様所有のキャッシュパラメータグループは、お客様がキャッシュパラメータグループを作成する度に作成されます。

お客様所有のキャッシュパラメータグループ:

- お客様が作成、所有します。
- お客様の互換性のあるいずれのクラスターにも割り当てることができます。
- カスタムキャッシュパラメータグループを作成するようにお客様が変更できます。

すべてのパラメータ値を変更できるわけではありません。詳細については、「[Redis 固有のパラメータ](#)」を参照してください。

パラメータグループを作成する

デフォルト値から変更するパラメータの値が 1 つ以上ある場合、新しいパラメータグループを作成する必要があります。ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、パラメータグループを作成できます。

パラメータグループを作成する (コンソール)

次の手順では、ElastiCache コンソールを使用してパラメータグループを編集する方法を示します。

ElastiCache コンソールを使用してパラメータグループを作成するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループを作成するには、[Create Parameter Group] を選択します。

[Create Parameter Group] 画面が表示されます。

4. [Family] のリストから、パラメータグループのテンプレートとなるパラメータグループファミリーを選択します。

や redis3.2 などのパラメータグループファミリーは、パラメータグループの実際のパラメータおよびその初期値を定義します。パラメータグループファミリーは、クラスターのエンジンおよびバージョンと一致している必要があります。

5. [Name] ボックスで、このパラメータグループの一意の名前を入力します。

クラスターを作成、またはクラスターのパラメータグループを変更するときは、パラメータグループを名前を選択します。したがって、わかりやすくパラメータグループのファミリーを特定するのに役立つ名前をお勧めします。

パラメータグループの命名に関する制約は次のとおりです。

- 先頭を ASCII 文字にする必要があります。
 - ASCII 文字、数字、ハイフンのみを含めることができます。
 - 1~255 文字にする必要があります。
 - 連続する 2 つのハイフンを含めることはできません。
 - ハイフンで終わることはできません。
6. [Description] ボックスに、パラメータグループの説明を入力します。
 7. パラメータグループを作成するには、[Create] を選択します。

パラメータグループを作成しないでプロセスを終了するには、[Cancel] を選択します。

8. パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「[パラメータグループを変更する](#)」を参照してください。

パラメータグループを作成する (AWS CLI)

AWS CLI を使用してパラメータグループを作成するには、以下のパラメータを指定して `create-cache-parameter-group` コマンドを使用します。

- `--cache-parameter-group-name` — パラメータグループの名前。

パラメータグループの命名に関する制約は次のとおりです。

- 先頭を ASCII 文字にする必要があります。
 - ASCII 文字、数字、ハイフンのみを含めることができます。
 - 1~255 文字にする必要があります。
 - 連続する 2 つのハイフンを含めることはできません。
 - ハイフンで終わることはできません。
- `--cache-parameter-group-family` — パラメータグループのエンジンとバージョンファミリー。
 - `--description` — パラメータグループについてユーザーが入力する説明。

Example

次の例では、redis2.8 ファミリーをテンプレートとして使用して、myRed28 という名前のパラメータグループを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-parameter-group \  
  --cache-parameter-group-name myRed28 \  
  --cache-parameter-group-family redis2.8 \  
  --description "My first parameter group"
```

Windows の場合:

```
aws elasticache create-cache-parameter-group ^  
  --cache-parameter-group-name myRed28 ^  
  --cache-parameter-group-family redis2.8 ^  
  --description "My first parameter group"
```

このコマンドの出力は次のようになります。

```
{  
  "CacheParameterGroup": {  
    "CacheParameterGroupName": "myRed28",  
    "CacheParameterGroupFamily": "redis2.8",  
    "Description": "My first parameter group"  
  }  
}
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「[パラメータグループを変更する](#)」を参照してください。

詳細については、「[create-cache-parameter-group](#)」を参照してください。

パラメータグループを作成する (ElastiCache API)

ElastiCache API を使用してパラメータグループを作成するには、以下のパラメータを指定して CreateCacheParameterGroup アクションを使用します。

- ParameterGroupName — パラメータグループの名前。

パラメータグループの命名に関する制約は次のとおりです。

- 先頭を ASCII 文字にする必要があります。
- ASCII 文字、数字、ハイフンのみを含めることができます。
- 1~255 文字にする必要があります。
- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。
- CacheParameterGroupFamily — パラメータグループのエンジンとバージョンファミリー。例えば、redis2.8 です。
- Description — パラメータグループについてユーザーが入力する説明。

Example

次の例では、redis2.8 ファミリーをテンプレートとして使用して、myRed28 という名前のパラメータグループを作成します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=CreateCacheParameterGroup  
&CacheParameterGroupFamily=redis2.8  
&CacheParameterGroupName=myRed28  
&Description=My%20first%20parameter%20group  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

このアクションからの応答は、次のようになります。

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazonaws.com/  
doc/2013-06-15/">  
  <CreateCacheParameterGroupResult>  
    <CacheParameterGroup>  
      <CacheParameterGroupName>myRed28</CacheParameterGroupName>  
      <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>  
      <Description>My first parameter group</Description>  
    </CacheParameterGroup>  
  </CreateCacheParameterGroupResult>  
</ResponseMetadata>
```

```
<RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
</ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「[パラメータグループを変更する](#)」を参照してください。

詳細については、「[CreateCacheParameterGroup](#)」を参照してください。

パラメータグループを名前別に一覧表示する

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、パラメータグループを一覧表示できます。

パラメータグループを名前別に一覧表示する (コンソール)

次の手順は、ElastiCache コンソールを使用してパラメータグループのリストを表示する方法を示します。

ElastiCache コンソールを使用してパラメータグループを一覧するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。

パラメータグループを名前別に一覧表示する (AWS CLI)

AWS CLI を使用してパラメータグループのリストを生成するには、`describe-cache-parameter-groups` コマンドを使用します。パラメータグループの名前を指定した場合は、そのパラメータグループのみが一覧表示されます。パラメータグループの名前を指定しない場合は、最大で `--max-records` のパラメータグループが一覧表示されます。いずれの場合も、パラメータグループの名前、ファミリー、および説明が表示されます。

Example

次のサンプルコードは、パラメータグループ `myRed28` のリストです。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameter-groups \  
  --cache-parameter-group-name myRed28
```

Windows の場合:

```
aws elasticache describe-cache-parameter-groups ^  
  --cache-parameter-group-name myRed28
```

このコマンドの出力は、名前の一覧、ファミリー、パラメータグループの説明となります。

```
{
  "CacheParameterGroups": [
    {
      "CacheParameterGroupName": "myRed28",
      "CacheParameterGroupFamily": "redis2.8",
      "Description": "My first parameter group"
    }
  ]
}
```

Example

次のサンプルコードは、Redis エンジンバージョン 5.0.6 以降で実行されているパラメータグループのパラメータグループ `myRed56` を示しています。パラメータグループが[グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)の一部である場合、出力で返される `IsGlobal` プロパティ値は `Yes` になります。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameter-groups \
  --cache-parameter-group-name myRed56
```

Windows の場合:

```
aws elasticache describe-cache-parameter-groups ^
  --cache-parameter-group-name myRed56
```

このコマンドの出力は、名前の一覧、ファミリー、isGlobal、パラメータグループの説明となります。

```
{
  "CacheParameterGroups": [
    {
      "CacheParameterGroupName": "myRed56",
      "CacheParameterGroupFamily": "redis5.0",
      "Description": "My first parameter group",
      "IsGlobal": "yes"
    }
  ]
}
```

Example

次のサンプルコードリストには、最大で 10 個のパラメータグループが一覧されています。

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

このコマンドの JSON 出力は、名前、ファミリー、説明を一覧表示し、redis5.6 の場合は、パラメータグループが Global Datastore の一部である (isGlobal) かどうかを各パラメータグループについて表示します。

```
{
  "CacheParameterGroups": [
    {
      "CacheParameterGroupName": "custom-redis32",
      "CacheParameterGroupFamily": "redis3.2",
      "Description": "custom parameter group with reserved-memory > 0"
    },
    {
      "CacheParameterGroupName": "default.memcached1.4",
      "CacheParameterGroupFamily": "memcached1.4",
      "Description": "Default parameter group for memcached1.4"
    },
    {
      "CacheParameterGroupName": "default.redis2.6",
      "CacheParameterGroupFamily": "redis2.6",
      "Description": "Default parameter group for redis2.6"
    },
    {
      "CacheParameterGroupName": "default.redis2.8",
      "CacheParameterGroupFamily": "redis2.8",
      "Description": "Default parameter group for redis2.8"
    },
    {
      "CacheParameterGroupName": "default.redis3.2",
      "CacheParameterGroupFamily": "redis3.2",
      "Description": "Default parameter group for redis3.2"
    },
    {
      "CacheParameterGroupName": "default.redis3.2.cluster.on",
      "CacheParameterGroupFamily": "redis3.2",
      "Description": "Customized default parameter group for redis3.2 with
cluster mode on"
    },
  ],
}
```

```
{
  "CacheParameterGroupName": "default.redis5.6.cluster.on",
  "CacheParameterGroupFamily": "redis5.0",
  "Description": "Customized default parameter group for redis5.6 with
cluster mode on",
  "isGlobal": "yes"
},
]
```

詳細については、「[describe-cache-parameter-groups](#)」を参照してください。

パラメータグループを名前別に一覧表示する (ElastiCache API)

ElastiCache API を使用してパラメータグループのリストを生成するには、DescribeCacheParameterGroups アクションを使用します。パラメータグループの名前を指定した場合は、そのパラメータグループのみが一覧表示されます。パラメータグループの名前を指定しない場合は、最大で MaxRecords のパラメータグループが一覧表示されます。いずれの場合も、パラメータグループの名前、ファミリー、および説明が表示されます。

Example

次のサンプルコードリストには、最大で 10 個のパラメータグループが一覧されています。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheParameterGroups
&MaxRecords=10
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

このアクションからの応答は、名前、ファミリー、説明を一覧表示し、redis5.6 の場合は、パラメータグループが Global Datastore に属している (isGlobal) かどうかを各パラメータグループについて説明します。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
```



```

    <CacheParameterGroupName>myRedis28</CacheParameterGroupName>
    <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
    <Description>My custom Redis 2.8 parameter group</Description>
  </CacheParameterGroup>
  <CacheParameterGroup>
    <CacheParameterGroupName>myMem14</CacheParameterGroupName>
    <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
    <Description>My custom Memcached 1.4 parameter group</Description>
  </CacheParameterGroup>
  <CacheParameterGroup>
    <CacheParameterGroupName>myRedis56</CacheParameterGroupName>
    <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
    <Description>My custom redis 5.6 parameter group</Description>
    <isGlobal>yes</isGlobal>
  </CacheParameterGroup>
</CacheParameterGroups>
</DescribeCacheParameterGroupsResult>
<ResponseMetadata>
  <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
</ResponseMetadata>
</DescribeCacheParameterGroupsResponse>

```

Example

次のサンプルコードは、パラメータグループ `myRed28` のリストです。

```

https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheParameterGroups
&CacheParameterGroupName=myRed28
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>

```

このアクションからの応答は、名前、ファミリー、説明となります。

```

<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed28</CacheParameterGroupName>

```

```

    <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
    <Description>My custom Redis 2.8 parameter group</Description>
  </CacheParameterGroup>
</CacheParameterGroups>
</DescribeCacheParameterGroupsResult>
<ResponseMetadata>
  <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
</ResponseMetadata>
</DescribeCacheParameterGroupsResponse>

```

Example

次のサンプルコードは、パラメータグループ `myRed56` のリストです。

```

https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheParameterGroups
&CacheParameterGroupName=myRed56
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>

```

このアクションからの応答は、名前、ファミリー、説明を一覧表示し、パラメータグループが Global Datastore の一部である (isGlobal) かどうかを表示します。

```

<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom Redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>

```

詳細については、「[DescribeCacheParameterGroups](#)」を参照してください。

パラメータグループの値を一覧表示する

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、パラメータグループのパラメータとその値を一覧表示できます。

パラメータグループの値を一覧表示する (コンソール)

次の手順は、ElastiCache コンソールを使用してパラメータグループのパラメータと値を一覧表示する方法を示しています。

ElastiCache コンソールを使用してパラメータグループのパラメータとその値を表示するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループ名の左側にあるボックスを選択して、パラメータと値を一覧表示するパラメータグループを選択します。

パラメータと値は画面の下部に表示されます。パラメータの数によっては、スクロールして関心のあるパラメータを検索する必要がある場合もあります。

パラメータグループの値を一覧表示する (AWS CLI)

AWS CLI を使用してパラメータグループのパラメータとその値の一覧を表示するには、`describe-cache-parameters` コマンドを使用します。

Example

次のサンプルコードは、パラメータグループ `myRedis28` のすべてのパラメータと値リストを一覧します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameters \  
  --cache-parameter-group-name myRedis28
```

Windows の場合:

```
aws elasticache describe-cache-parameters ^
```

```
--cache-parameter-group-name myRed28
```

詳細については、「[describe-cache-parameters](#)」を参照してください。

パラメータグループの値を一覧表示する (ElastiCache API)

ElastiCacheAPI を使用してパラメータグループのパラメータとその値を一覧表示するには、DescribeCacheParameters アクションを使用します。

Example

次のサンプルコードは、パラメータグループ *myRed28* のすべてのパラメータを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheParameters  
&CacheParameterGroupName=myRed28  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

このアクションからの応答は、次のようになります。この応答には短縮されています。

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/  
doc/2013-06-15/">  
  <DescribeCacheParametersResult>  
    <CacheClusterClassSpecificParameters>  
      <CacheNodeTypeSpecificParameter>  
        <DataType>integer</DataType>  
        <Source>system</Source>  
        <IsModifiable>>false</IsModifiable>  
        <Description>The maximum configurable amount of memory to use to store items,  
in megabytes.</Description>  
        <CacheNodeTypeSpecificValues>  
          <CacheNodeTypeSpecificValue>  
            <Value>1000</Value>  
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>  
          </CacheNodeTypeSpecificValue>  
          <CacheNodeTypeSpecificValue>  
            <Value>6000</Value>  
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
```

```
</CacheNodeTypeSpecificValue>
<CacheNodeTypeSpecificValue>
  <Value>7100</Value>
  <CacheClusterClass>cache.m1.large</CacheClusterClass>
</CacheNodeTypeSpecificValue>
<CacheNodeTypeSpecificValue>
  <Value>1300</Value>
  <CacheClusterClass>cache.m1.small</CacheClusterClass>
</CacheNodeTypeSpecificValue>

...output omitted...

</CacheClusterClassSpecificParameters>
</DescribeCacheParametersResult>
<ResponseMetadata>
  <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
</ResponseMetadata>
</DescribeCacheParametersResponse>
```

詳細については、「[DescribeCacheParameters](#)」を参照してください。

パラメータグループを変更する

Important

デフォルトのパラメータグループを変更することはできません。

パラメータグループでいくつかのパラメータを変更できます。これらのパラメータ値は、パラメータグループに関連付けられるクラスターに適用されます。パラメータ値の変更がパラメータグループに適用される場合の詳細については、「[Redis 固有のパラメータ](#)」を参照してください。

パラメータグループを変更する (コンソール)

次の手順では、ElastiCache コンソールで cluster-enabled パラメータ値を変更する方法を説明します。同じ手順を使用して、すべてのパラメータを変更します。

ElastiCache コンソールを使用してパラメータ値を変更するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。

2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループ名の左側にあるボックスを選択して、変更する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パラメータは、画面の下部に表示されます。すべてのパラメータを確認するには、ページでリストを作成する必要があります。

4. 複数のパラメータを修正するには、[Edit Parameters] を選択します。
5. [Save changes] (変更の保存) をクリックします。
6. 変更したパラメータの名前を検索するには、「[Redis 固有のパラメータ](#)」を参照してください。Redis (クラスターモードが無効) クラスターがあり、以下のパラメータを変更する場合は、クラスター内のノードを再起動する必要があります。

- アクティブハッシュ化
- データベース

詳細については、「[ノードの再起動](#)」を参照してください。

Redis (クラスターモードが有効) パラメータの変更

Redis (クラスターモードが有効) クラスターで次のパラメータを変更する場合は、以降のステップに従います。

- アクティブハッシュ化
- データベース

1. クラスターの手動バックアップを作成します。「[手動バックアップの取得](#)」を参照してください。
2. Redis (クラスターモードが有効) クラスターを削除します。「[クラスターの削除](#)」を参照してください。
3. 変更されたパラメータグループとバックアップを使用してクラスターを復元し、新しいクラスターをシードします。「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

他のパラメータを変更する場合、これは必要ありません。

パラメータグループを変更する (AWS CLI)

AWS CLI を使用してパラメータの値を変更するには、`modify-cache-parameter-group` コマンドを使用します。

Example

変更するパラメータの名前と許容値を検索するには、「[Redis 固有のパラメータ](#)」を参照してください。

次のサンプルコードでは、パラメータグループ `myredis32-on-30` で `[reserved-memory-percent]` と `[cluster-enabled]` の 2 つのパラメータの値を設定します。ここでは、`[reserved-memory-percent]` を 30 (30 パーセント) に、`[cluster-enabled]` を `yes` に設定します。これでパラメータグループは Redis (クラスターモードが有効) クラスター (レプリケーショングループ) で使用できるようになります。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-parameter-group \  
  --cache-parameter-group-name myredis32-on-30 \  
  --parameter-name-values \  
    ParameterName=reserved-memory-percent,ParameterValue=30 \  
    ParameterName=cluster-enabled,ParameterValue=yes
```

Windows の場合:

```
aws elasticache modify-cache-parameter-group ^  
  --cache-parameter-group-name myredis32-on-30 ^  
  --parameter-name-values ^  
    ParameterName=reserved-memory-percent,ParameterValue=30 ^  
    ParameterName=cluster-enabled,ParameterValue=yes
```

このコマンドの出力は次のようになります。

```
{  
  "CacheParameterGroupName": "my-redis32-on-30"  
}
```

詳細については、「[modify-cache-parameter-group](#)」を参照してください。

変更したパラメータの名前を検索するには、「[Redis 固有のパラメータ](#)」を参照してください。

Redis (クラスターモードが無効) クラスターがあり、以下のパラメータを変更する場合は、クラスター内のノードを再起動する必要があります。

- アクティブハッシュ化
- データベース

詳細については、「[ノードの再起動](#)」を参照してください。

i Redis (クラスターモードが有効) パラメータの変更

Redis (クラスターモードが有効) クラスターで次のパラメータを変更する場合は、以降のステップに従います。

- アクティブハッシュ化
- データベース

1. クラスターの手動バックアップを作成します。「[手動バックアップの取得](#)」を参照してください。
2. Redis (クラスターモードが有効) クラスターを削除します。「[クラスターの削除](#)」を参照してください。
3. 変更されたパラメータグループとバックアップを使用してクラスターを復元し、新しいクラスターをシードします。「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

他のパラメータを変更する場合、これは必要ありません。

パラメータグループを変更する (ElastiCache API)

ElastiCache API を使用してパラメータグループのパラメータ値を変更するには、ModifyCacheParameterGroup アクションを使用します。

Example

変更するパラメータの名前と許容値を検索するには、「[Redis 固有のパラメータ](#)」を参照してください。

次のサンプルコードでは、パラメータグループ `myredis32-on-30` で `[reserved-memory-percent]` と `[cluster-enabled]` の 2 つのパラメータの値を設定します。ここでは、`[reserved-memory-percent]` を 30 (30 パーセント) に、`[cluster-enabled]` を `yes` に設定します。これでパラメータグループは Redis (クラスターモードが有効) クラスター (レプリケーショングループ) で使用できるようになります。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyCacheParameterGroup  
&CacheParameterGroupName=myredis32-on-30  
&ParameterNameValues.member.1.ParameterName=reserved-memory-percent  
&ParameterNameValues.member.1.ParameterValue=30  
&ParameterNameValues.member.2.ParameterName=cluster-enabled  
&ParameterNameValues.member.2.ParameterValue=yes  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

詳細については、「[ModifyCacheParameterGroup](#)」を参照してください。

Redis (クラスターモードが無効) クラスターがあり、以下のパラメータを変更する場合は、クラスター内のノードを再起動する必要があります。

- アクティブハッシュ化
- データベース

詳細については、「[ノードの再起動](#)」を参照してください。

Redis (クラスターモードが有効) パラメータの変更

Redis (クラスターモードが有効) クラスターで次のパラメータを変更する場合は、以降のステップに従います。

- アクティブハッシュ化
- データベース

1. クラスターの手動バックアップを作成します。「[手動バックアップの取得](#)」を参照してください。

2. Redis (クラスターモードが有効) クラスターを削除します。「[クラスターの削除](#)」を参照してください。
3. 変更されたパラメータグループとバックアップを使用してクラスターを復元し、新しいクラスターをシードします。「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

他のパラメータを変更する場合、これは必要ありません。

パラメータグループを削除する

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、カスタムパラメータグループを削除できます。

パラメータグループがクラスターに関連付けられている場合は、パラメータグループを削除できません。デフォルトのパラメータグループも削除できません。

パラメータグループを削除する (コンソール)

次の手順では、ElastiCache コンソールを使用してパラメータグループを削除する方法を示します。

ElastiCache コンソールを使用してパラメータグループを削除するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループ名の左側にあるボックスを選択して、削除するパラメータグループを選択します。

[Delete] ボタンがアクティブになります。

4. [Delete] (削除) をクリックします。

[Delete Parameter Groups] の確認画面が表示されます。

5. パラメータグループを削除するには、[Delete Parameter Groups] の確認画面で [Delete] を選択します。

パラメータグループを保持するには、[Cancel] を選択します。

パラメータグループを削除する (AWS CLI)

AWS CLI を使用してパラメータグループを削除するには、`delete-cache-parameter-group` コマンドを使用します。削除するパラメータグループで、`--cache-parameter-group-name` で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

次のサンプルコードは、`myMem14` パラメータグループを削除します。

Example

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-parameter-group \  
  --cache-parameter-group-name myRed28
```

Windows の場合:

```
aws elasticache delete-cache-parameter-group ^  
  --cache-parameter-group-name myRed28
```

詳細については、「[delete-cache-parameter-group](#)」を参照してください。

パラメータグループを削除する (ElastiCache API)

ElastiCache API を使用してパラメータグループを削除するには、DeleteCacheParameterGroup アクションを使用します。削除するパラメータグループで、CacheParameterGroupName で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

Example

次のサンプルコードは、myRed28 パラメータグループを削除します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DeleteCacheParameterGroup  
&CacheParameterGroupName=myRed28  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

詳細については、「[DeleteCacheParameterGroup](#)」を参照してください。

Memcached 固有のパラメータ

Memcached クラスターにパラメータグループを指定しない場合、エンジンのバージョンに適したデフォルトのパラメータグループが使用されます。デフォルトのパラメータグループのパラメータの値を変更することはできません。ただし、カスタムパラメータグループを作成し、いつでもクラスターに割り当てることはできます。詳細については、「[パラメータグループを作成する](#)」を参照してください。

トピック

- [Memcached 1.6.17 の変更点](#)
- [Memcached 1.6.6 で追加されたパラメータ](#)
- [Memcached 1.5.10 パラメータの変更](#)
- [Memcached 1.4.34 で追加されたパラメータ](#)
- [Memcached 1.4.33 で追加されたパラメータ](#)
- [Memcached 1.4.24 で追加されたパラメータ](#)
- [Memcached 1.4.14 で追加されたパラメータ](#)
- [Memcached 1.4.5 でサポートされているパラメータ](#)
- [Memcached 接続オーバーヘッド](#)
- [Memcached ノードタイプ固有のパラメータ](#)

Memcached 1.6.17 の変更点

Memcached 1.6.17 以降、`lru_crawler`、`lru`、および `slabs` 管理コマンドはサポートされなくなりました。これらの変更により、`lru_crawler` コマンドを使ってランタイムで有効または無効にできなくなります。`lru_crawler` は、カスタムパラメータグループを変更して有効または無効にしてください。

Memcached 1.6.6 で追加されたパラメータ


Memcached 1.6.6 では、追加のパラメータはサポートされません。

パラメータグループファミリー: `memcached1.6`

Memcached 1.5.10 パラメータの変更

Memcached 1.5.10 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: `memcached1.5`

名前	詳細	説明
no_modern	デフォルト: 1 タイプ: ブール値 変更可能: はい 許可された値: 0,1 変更の適用: 起動時	<p>slab_reassign、slab_auto move、lru_crawler、lru_maintainer、maxconns_fast コマンドを無効にするためのエイリアス。No modern は、hash_algorithm も jenkins に設定し、ASCII VALUE のインラインも許可します。Memcached 1.5 以上に適用されます。最新に戻すには、このパラメータを無効にして再起動する必要があります。これにより、slab_reassign、slab_auto move、lru_crawler、lru_maintainer、および maxconns_fast が自動的に有効になります。</p> <div data-bbox="1008 1224 1508 1879" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>2021 年 8 月 20 日現在、このパラメータのデフォルトの設定値は 0 から 1 に変更されています。更新されたデフォルト値は、2021 年 8 月 20 日以降、各リージョンの Elasticache の新規ユーザーによって自動的に取得されます。2021 年 8 月 20 日以前のリージョンで ElastiCache を使用する既</p> </div>

名前	詳細	説明
		<p>存ユーザーは、この新しい変更を取得するためにカスタムパラメータグループを手動で変更する必要があります。</p>
inline_ascii_resp	デフォルト: 0 タイプ: ブール値 変更可能: はい 許可された値: 0,1 変更の適用: 起動時	アイテム内の VALUE レスポンスからの数値が保存されます。最大 24 バイトを使用します。ASCII get、faster セットの小さい減速。

Memcached 1.5.10 では、次のパラメータが削除されます。

名前	詳細	説明
expirezero_does_no_t_evict	デフォルト: 0 タイプ: ブール値 変更可能: はい 許可された値: 0,1 変更の適用: 起動時	このバージョンではサポートされなくなりました。
modern	デフォルト: 1 タイプ: ブール値 変更可能: はい (no_modern に設定)	このバージョンではサポートされなくなりました。このバージョン以降、起動または再起動するたびに no-modern がデフォルトで有効になります。

名前	詳細	説明
	されている場合は再起動が必要です) 許可された値: 0,1 変更の適用: 起動時	

Memcached 1.4.34 で追加されたパラメータ

Memcached 1.4.34 では、追加のパラメータはサポートされません。

パラメータグループファミリー: memcached1.4

Memcached 1.4.33 で追加されたパラメータ

Memcached 1.4.33 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: memcached1.4

名前	詳細	説明
modern	デフォルト: 有効 タイプ: ブール値 変更可能: はい 変更の適用: 起動時	各種機能のエイリアス有効化 modern は、次のコマンドをオンにし、murmur3 ハッシュアルゴリズム (slab_reassign、slab_auto move、lru_crawler、lru_maintainer、maxconns_fast、hash_algorithm=murmur3) を使用する場合と同等です。
watch	デフォルト: 有効 タイプ: ブール値	ログ取得、削除または変異。たとえば、watch をオンにすると、get、set、delete または

名前	詳細	説明
	変更可能: はい 変更の適用: 即時 ログは、 <code>watcher_logbuf_size</code> および <code>worker_logbuf_size</code> 制限に達すると削除できます。	<code>update</code> が発生したときにユーザーはログを表示できます。
<code>idle_timeout</code>	デフォルト: 0 (無効) タイプ: 整数 変更可能: はい 変更の適用: 起動時	閉じる前にクライアントがアイドル状態にできる最小秒数。値の範囲: 0 ~ 86400
<code>track_sizes</code>	デフォルト: 無効 タイプ: ブール値 変更可能: はい 変更の適用: 起動時	各スラブグループの消費サイズを表示します。 有効化 <code>track_sizes</code> を行うと、 <code>stats sizes</code> を実行せずに <code>stats sizes_enable</code> を実行できます。

名前	詳細	説明
watcher_logbuf_size	デフォルト: 256 (KB) タイプ: 整数 変更可能: はい 変更の適用: 起動時	watch コマンドは、Memcached の配信ログ作成をオンにします。ただし、削除、変異、取得によって、ロギングバッファがいっぱいになる可能性がある場合には、watch でログを削除することができます。このような場合、ユーザーは、バッファサイズを増やして、ログ損失の可能性を抑えることができます。
worker_logbuf_size	デフォルト: 64 (KB) タイプ: 整数 変更可能: はい 変更の適用: 起動時	watch コマンドは、Memcached の配信ログ作成をオンにします。ただし、削除、変異、取得によって、ロギングバッファがいっぱいになる可能性がある場合には、watch でログを削除することができます。このような場合、ユーザーは、バッファサイズを増やして、ログ損失の可能性を抑えることができます。
slab_chunk_max	デフォルト: 524288 (バイト) タイプ: 整数 変更可能: はい 変更の適用: 起動時	スラブの最大サイズを指定します。スラブサイズを小さくすると、メモリは効率的に使用されません。slab_chunk_max より大きい項目は、複数のスラブに分割されます。

名前	詳細	説明
lru_crawler metadump [all 1 2 3]	デフォルト: 無効 タイプ: ブール値 変更可能: はい 変更の適用: 即時	lru_crawler を有効化すると、このコマンドによってすべてのキーがダンプされます。 all 1 2 3 - すべてのスラブ、または特定のスラブ数を指定する

Memcached 1.4.24 で追加されたパラメータ

Memcached 1.4.24 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: memcached1.4

名前	詳細	説明
disable_flush_all	デフォルト: 0 (無効) タイプ: ブール値 変更可能: はい 変更の適用: 起動時	flush_all を無効化するパラメータ (-F) を追加します。本稼働インスタンスでフルフラッシュを実行しない場合に便利です。 値: 0、1 (値が 0 の場合にユーザーは flush_all を実行できません。)
hash_algorithm	デフォルト: jenkins タイプ: 文字列 変更可能: はい 変更の適用: 起動時	使用されるハッシュアルゴリズム。使用可能な値: murmur3 と jenkins。
lru_crawler	デフォルト: 0 (無効) タイプ: ブール値	期限が切れた項目のスラブクラスを消去します。これにより、バックグラウンドで実行されるプロ

名前	詳細	説明
	<p>変更可能: はい</p> <p>変更の適用: 再起動後</p> <div data-bbox="651 367 971 1066" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>実行時に、コマンドラインから <code>lru_crawler</code> を一時的に有効にすることができます。詳細については、「Describe」列を参照してください。</p> </div>	<p>セスの影響を小さくなります。現在は、手動コマンドを使用して <code>Crawl</code> を起動する必要があります。</p> <p>一時的に有効にするには、コマンドラインで <code>lru_crawler enable</code> を実行します。</p> <p><code>lru_crawler 1,3,5</code> はスラブクラス 1、3、5 をクロールし、<code>freelist</code> に追加する期限切れの項目を検索します。</p> <p>値: 0、1</p> <div data-bbox="1008 909 1507 1556" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>コマンドラインで <code>lru_crawler</code> を有効にして、コマンドラインまたは次の再起動で無効化されるまでクローラを有効にします。永続的に有効にするには、パラメータ値を変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。</p> </div>

名前	詳細	説明
<code>lru_maintainer</code>	デフォルト: 0 (無効) タイプ: ブール値 変更可能: はい 変更の適用: 起動時	容量に到達すると LRU 間で項目をシャッフルするバックグラウンドスレッドです。値: 0、1。
<code>expirezero_does_no_t_evict</code>	デフォルト: 0 (無効) タイプ: ブール値 変更可能: はい 変更の適用: 起動時	<p><code>lru_maintainer</code> と併用すると、項目の期限切れ時間が 0 (期限切れなし) になります。</p> <div style="border: 1px solid #f08080; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p>⚠ Warning</p> <p>これにより、期限切れでクリアされる他の項目をメモリから排除して、メモリを使用できるようにすることができます。</p> </div> <p><code>lru_maintainer</code> を無視するよう設定できます。</p>

Memcached 1.4.14 で追加されたパラメータ

Memcached 1.4.14 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: memcached1.4

Memcached 1.4.14 で追加されたパラメータ

名前	説明
config_max	ElastiCache 設定エントリの最大数。
config_size_max	設定エントリの最大サイズ (バイト単位)。

名前	説明
hashpower_init	ElastiCache ハッシュテーブルの初期サイズは、2 の累乗として表されます。デフォルトは 16 (2^{16})、つまり 65536 のキーです。

名前	説明
maxconns_fast	<p>最大接続制限に達したときに新しい接続リクエストを処理する方法を変更します。このパラメータを 0 (ゼロ) に設定した場合、新しい接続がバックログキューに追加され、他の接続が終了するまで待機します。パラメータを 1 に設定した場合、ElastiCache はクライアントにエラーを送信し、すぐに接続を終了します。</p>

名前	説明
slab_automove	<p>スラブ自動移動アルゴリズムを調整します。このパラメータを 0 (ゼロ) に設定した場合、自動移動アルゴリズムは無効です。1 に設定した場合、ElastiCache は低速で控えめな手法を使用して、スラブを自動的に移動します。2 に設定した場合、削除が生じると必ず ElastiCache はスラブを積極的に移動します (このモードは、テスト目的以外では推奨されません)。</p>

名前	説明
slab_reassign	スラブの再割り当てを有効または無効にします。このパラメータを 1 に設定した場合、「slabs reassign」コマンドを使用してメモリを手動で再割り当てできます。

Memcached 1.4.5 でサポートされているパラメータ

パラメータグループファミリー: memcached1.4

Memcached 1.4.5 では、さらに次のパラメータがサポートされています。

Memcached 1.4.5 で追加されたパラメータ

名前	詳細	説明
backlog_queue_limit	デフォルト: 1024 タイプ: 整数 変更可能: いいえ	バックログキューの制限。
binding_protocol	デフォルト: auto タイプ: 文字列 変更可能: はい 変更の適用: 再起動後	バインディングプロトコル。 許可される値は ascii および auto です。 binding_protocol の値を変更する際のガイダンスについては、「 パラメータグループを変更する 」を参照してください。
cas_disabled	デフォルト: 0 (false) 型: ブール 変更可能: はい 変更の適用: 再起動後	1 (true) の場合、CAS (Check and Set) 操作が無効になり、格納されている項目が消費するバイト数は CAS が有効な場合より 8 バイト少なくなります。
chunk_size	デフォルト: 48 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	最も小さい項目のキー、値、およびフラグ (バイト単位) に割り当てる領域の最小量 (バイト単位)。
chunk_size_growth_factor	デフォルト: 1.25 タイプ: 浮動小数点 変更可能: はい 変更の適用: 再起動後	連続する各 memcached チャンクのサイズを制御する増加係数。各チャンクは、前のチャンクより chunk_size_growth_factor 倍大きくなります。

名前	詳細	説明
error_on_memory_exhausted	デフォルト: 0 (false) 型: ブール 変更可能: はい 変更の適用: 再起動後	1 (true) の場合、項目を保存するメモリがないと、Memcached によって項目が削除されるのではなくエラーが返されます。
large_memory_pages	デフォルト: 0 (false) 型: ブール 変更可能: いいえ	1 (true) の場合、ElastiCache は大量のメモリページを使用しようとします。
lock_down_paged_memory	デフォルト: 0 (false) 型: ブール 変更可能: いいえ	1 (true) の場合、ElastiCache はすべてのページ分割メモリをロックダウンします。
max_item_size	デフォルト: 1048576 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	クラスターに保存できる最も大きい項目のサイズ (バイト単位)。
max_simultaneous_connections	デフォルト: 65000 タイプ: 整数 変更可能: いいえ	同時接続の最大数。
maximize_core_file_limit	デフォルト: 0 (false) 型: ブール 変更可能: 変更の適用: 再起動後	1 (true) の場合、ElastiCache はコアファイルの制限を最大限に高くします。

名前	詳細	説明
memcached_connections_overhead	デフォルト: 100 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	Memcached 接続および他のさまざまなオーバーヘッド用に予約されるメモリの量。このパラメータの詳細については、「 Memcached 接続オーバーヘッド 」を参照してください。
requests_per_event	デフォルト: 20 タイプ: 整数 変更可能: いいえ	特定の接続のイベントごとの最大リクエスト数。この制限は、リソース不足を防ぐために必要です。

Memcached 接続オーバーヘッド

各ノードで、項目の保存に使用可能なメモリは、ノード上の使用可能な合計メモリ (max_cache_memory パラメータ内) から、接続や他のオーバーヘッドに使用されているメモリ (memcached_connections_overhead パラメータ内) を引いた量です。たとえば、タイプが cache.m1.small のノードには 1300MB の max_cache_memory があります。memcached_connections_overhead がデフォルト値の 100 MB の場合、Memcached プロセスは項目を保存するために 1,200 MB 使用できます。

memcached_connections_overhead パラメータのデフォルト値は、ほとんどのユースケースに適しています。ただし、接続オーバーヘッドの割り当てに必要な量は、リクエストの頻度、ペイロードサイズ、接続数など、複数の要因によって変化します。

アプリケーションのニーズにさらに合うように memcached_connections_overhead の値を変更できます。たとえば、memcached_connections_overhead パラメータの値を大きくすると、項目の保存に使用できるメモリの量が減り、接続のオーバーヘッド用のバッファが増えます。memcached_connections_overhead パラメータの値を小さくすると、項目の保存に使用できるメモリは増えますが、スワップの使用とパフォーマンスの低下のリスクが高くなります。スワップの使用やパフォーマンスの低下が観察される場合、memcached_connections_overhead パラメータの値を大きくしてみてください。

⚠ Important

ノードタイプが `cache.t1.micro` の場合、`memcached_connections_overhead` の値は次のように決まります。

- クラスターがデフォルトのパラメータグループを使用している場合、ElastiCache は `memcached_connections_overhead` の値を 13 MB に設定します。
- 自身で作成したパラメータグループをクラスターが使用している場合、`memcached_connections_overhead` の値を選択した値に設定できます。

Memcached ノードタイプ固有のパラメータ

ほとんどのパラメータの値は 1 つですが、一部のパラメータには、使用されているノードタイプによって複数の値が設定されることがあります。次の表は、各ノードタイプの `max_cache_memory` パラメータと `num_threads` パラメータのデフォルト値を示しています。これらのパラメータの値は変更できません。

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.t1.micro	213	1
cache.t2.micro	555	1
cache.t2.small	1588	1
cache.t2.medium	3301	2
cache.t3.micro	512	2
cache.t3.small	1402	2
cache.t3.medium	3364	2
cache.t4g.micro	512	2
cache.t4g.small	1402	2
cache.t4g.medium	3164	2

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.m1.small	1301	1
cache.m1.medium	3350	1
cache.m1.large	7100	2
cache.m1.xlarge	14600	4
cache.m2.xlarge	33800	2
cache.m2.2xlarge	30412	4
cache.m2.4xlarge	68000	16
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.m4.large	6573	2
cache.m4.xlarge	11496	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
cache.m5.large	6537	2
cache.m5.xlarge	13248	4
cache.m5.2xlarge	26671	8
cache.m5.4xlarge	53516	16

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.m5.12xlarge	160900	48
cache.m5.24xlarge	321865	96
cache.m6g.large	6537	2
cache.m6g.xlarge	13248	4
cache.m6g.2xlarge	26671	8
cache.m6g.4xlarge	53516	16
cache.m6g.8xlarge	107000	32
cache.m6g.12xlarge	160900	48
cache.m6g.16xlarge	214577	64
cache.c1.xlarge	6600	8
cache.r3.large	13800	2
cache.r3.xlarge	29100	4
cache.r3.2xlarge	59600	8
cache.r3.4xlarge	120600	16
cache.r3.8xlarge	120600	32
cache.r4.large	12590	2
cache.r4.xlarge	25652	4
cache.r4.2xlarge	51686	8
cache.r4.4xlarge	103815	16
cache.r4.8xlarge	208144	32

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.r4.16xlarge	416776	64
cache.r5.large	13387	2
cache.r5.xlarge	26953	4
cache.r5.2xlarge	54084	8
cache.r5.4xlarge	108347	16
cache.r5.12xlarge	325400	48
cache.r5.24xlarge	650869	96
cache.r6g.large	13387	2
cache.r6g.xlarge	26953	4
cache.r6g.2xlarge	54084	8
cache.r6g.4xlarge	108347	16
cache.r6g.8xlarge	214577	32
cache.r6g.12xlarge	325400	48
cache.r6g.16xlarge	429154	64
cache.c7gn.large	3164	2
cache.c7gn.xlarge	6537	4
cache.c7gn.2xlarge	13248	8
cache.c7gn.4xlarge	26671	16
cache.c7gn.8xlarge	53516	32
cache.c7gn.12xlarge	325400	48

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.c7gn.16xlarge	108347	64

Note

すべての T2 インスタンスは、Amazon Virtual Private Cloud (Amazon VPC) で作成されます。

Redis 固有のパラメータ

Redis クラスターにパラメータグループを指定しない場合、エンジンのバージョンに適したデフォルトのパラメータグループが使用されます。デフォルトのパラメータグループのパラメータの値を変更することはできません。しかし、カスタムパラメータグループを作成し、いつでもクラスターに割り当てることができます。ただし、条件付きで変更可能なパラメータの値が両方のパラメータグループで同じである場合に限ります。詳細については、「[パラメータグループを作成する](#)」を参照してください。

トピック

- [Redis 7 パラメータの変更](#)
- [Redis 6.x パラメータの変更](#)
- [Redis 5.0.3 パラメータの変更](#)
- [Redis 5.0.0 パラメータの変更](#)
- [Redis 4.0.10 パラメータの変更](#)
- [Redis 3.2.10 パラメータの変更](#)
- [Redis 3.2.6 パラメータの変更](#)
- [Redis 3.2.4 パラメータの変更](#)
- [Redis 2.8.24 \(拡張\) で追加されたパラメータ](#)
- [Redis 2.8.23 \(拡張\) で追加されたパラメータ](#)
- [Redis 2.8.22 \(拡張\) で追加されたパラメータ](#)
- [Redis 2.8.21 で追加されたパラメータ](#)
- [Redis 2.8.19 で追加されたパラメータ](#)
- [Redis 2.8.6 で追加されたパラメータ](#)
- [Redis 2.6.13 パラメータ](#)
- [Redis のノードタイプ固有のパラメータ](#)

Redis 7 パラメータの変更

パラメータグループファミリー: redis7

Redis 7 のデフォルトのパラメータグループは次のとおりです。

- `default.redis7` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが無効) クラスターおよびレプリケーショングループに使用します。

- `default.redis7.cluster.on` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが有効) クラスターおよびレプリケーショングループに使用します。

Redis 7 で追加されたパラメータは次のとおりです。

名前	詳細	説明
<code>cluster-allow-pubsubshard-when-down</code>	<p>許可される値: <code>yes</code>、<code>no</code></p> <p>デフォルト: <code>yes</code></p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の適用: クラスター内のすべてのノードにわたって即時</p>	<p>デフォルトの <code>[yes]</code> に設定すると、クラスターがダウン状態でも、自分がスロットを所有しているとみなしている限り、ノードは pubsub シャードトラフィックを処理できます。</p>
<code>cluster-preferred-endpoint-type</code>	<p>許可される値: <code>ip</code>、<code>tls-dynamic</code></p> <p>デフォルト: <code>tls-dynamic</code></p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の適用: クラスター内のすべてのノードにわたって即時</p>	<p>この値は、MOVED/ASKING リクエストに対して返されるエンドポイントと、CLUSTER SLOTS および CLUSTER SHARDS のエンドポイントフィールドを制御します。値が IP に設定されると、ノードは IP アドレスをアドバタイズします。値が <code>tls-dynamic</code> に設定されている場合、ノードは <code>encryption-in-transit</code> が有効になっているとホスト名をアドバタイズし、それ以外の場合は <code>ip</code> アドレスをアドバタイズします。</p>
<code>latency-tracking</code>	<p>許可される値: <code>yes</code>、<code>no</code></p> <p>デフォルト: <code>no</code></p> <p>タイプ: 文字列</p> <p>変更可能: はい</p>	<p><code>[yes]</code> に設定すると、コマンドごとのレイテンシーが追跡され、INFO レイテンシー統計コマンドを使用してパーセンタイル分布をエクスポートし、LATENCY コマンドを使用して累積レイテンシー分布 (ヒストグラム) をエクスポートできます。</p>

名前	詳細	説明
	変更の適用: クラスター内のすべてのノードにわたって即時	
hash-max-listpack-entries	許可される値: 0+ デフォルト: 512 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためのハッシュエントリの最大数。
hash-max-listpack-value	許可される値: 0+ デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するための最大ハッシュエントリのしきい値。
zset-max-listpack-entries	許可される値: 0+ デフォルト: 128 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためにソートされたセットエントリの最大数。

名前	詳細	説明
zset-max-listpack-value	許可される値: 0+ デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためにソートされたセットエントリの最大しきい値。

Redis 7 で変更されたパラメータは次のとおりです。

名前	詳細	説明
activeresharding	変更可能: no。Redis 7 では、このパラメータはデフォルトで非表示および有効になっています。無効にするには、 サポートケース を作成する必要があります。	変更可能は Yes でした。

Redis 7 で削除されたパラメータは次のとおりです。

名前	詳細	説明
hash-max-ziplist-entries	許可される値: 0+ デフォルト: 512 タイプ: 整数 変更可能: はい	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用する

名前	詳細	説明
	変更の適用: クラスター内のすべてのノードにわたって即時	
hash-max-ziplist-value	許可される値: 0+ デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用する
zset-max-ziplist-entries	許可される値: 0+ デフォルト: 128 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用します。
zset-max-ziplist-value	許可される値: 0+ デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用します。

名前	詳細	説明
list-max-ziplist-size	許可される値: デフォルト: -2 タイプ: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	内部リストノードごとに許可されるエントリ数。

Redis 6.x パラメータの変更

パラメータグループファミリー: redis6.x

Redis 6.x のデフォルトのパラメータグループは次のとおりです。

- `default.redis6.x` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが無効) クラスターおよびレプリケーショングループに使用します。
- `default.redis6.x.cluster.on` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが有効) クラスターおよびレプリケーショングループに使用します。

Note

Redis エンジンバージョン 6.2 で、[データ階層化](#) で使用するための r6gd ノードファミリーが導入された場合、r6gd ノードタイプでは、noeviction、volatile-lru、および allkeys-lru の max-memory ポリシーのみがサポートされます。

詳細については、「[ElastiCache for Redis バージョン 6.2 \(拡張\)](#)」および「[ElastiCache for Redis バージョン 6.0 \(拡張\)](#)」を参照してください。

Redis 6.x で追加されたパラメータは次のとおりです。。

名前	詳細	説明
acl-pubsub-default (added in 6.2)	<p>許可される値: resetchannels、allchannels</p> <p>デフォルト: allchannels</p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の適用: クラスターに関連付けられている既存の Redis ユーザーには、既存のアクセス許可が引き続き付与されます。ユーザーを更新するか、クラスターを再起動して、既存の Redis ユーザーを更新します。</p>	<p>このクラスターにデプロイした ACL ユーザーの、デフォルトの pubsub チャンネルのアクセス許可。</p>
cluster-allow-reads-when-down (added in 6.0)	<p>デフォルト: いいえ</p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の適用: クラスター内のすべてのノードにわたって即時</p>	<p>yes に設定すると、Redis (クラスターモードが有効) レプリケーショングループは、ノードがプライマリキーのクォーラムに到達できない場合でも、読み取りコマンドを処理し続けます。</p> <p>デフォルトの no に設定すると、レプリケーショングループはすべてのコマンドを拒否します。ノードグループが 3 つ未満のクラスターを使用している場合、またはアプリケーションで古い読み取りを安全に処理できる場合は、この値を yes に設定することをお勧めします。</p>
tracking-table-max-keys (added in 6.0)	<p>デフォルト: 1,000,000</p> <p>タイプ: 数値</p> <p>変更可能: はい</p>	<p>クライアント側のキャッシュを支援するために、Redis では、どのクライアントがどのキーにアクセスしたかの追跡をサポートします。</p> <p>追跡されたキーが変更されると、無効化メッセージがすべてのクライアントに送信され、</p>

名前	詳細	説明
	変更の適用: クラスター内のすべてのノードにわたって即時	キャッシュされた値が無効になったことが通知されます。この値により、このテーブルの上限を指定できます。このパラメータ値を超えると、クライアントには無作為に無効化が送信されます。この値は、十分なキーを追跡し続けながら、メモリ使用量を制限するように調整する必要があります。キーはメモリ不足状態でも無効になります。
acllog-max-len (added in 6.0)	デフォルト: 128 タイプ: 数値 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	この値は、ACL ログ内のエントリの最大数に対応します。

名前	詳細	説明
active-expire-effort (added in 6.0)	デフォルト: 1 タイプ: 数値 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	<p>Redis は、2 つのメカニズムによって、有効期限 (TTL) を超えたキーを削除します。1 つでは、キーがアクセスされ、期限切れであることが判明します。もう 1 つでは、定期的なジョブがキーをサンプリングし、有効期限 (TTL) を超えたキーを期限切れにします。このパラメータは、Redis が定期ジョブ内のアイテムを期限切れにするために使用する作業量を定義します。</p> <p>デフォルト値の 1 では、期限切れのキーの 10% 以上をメモリに残さないようにします。また、合計メモリの 25% 以上を消費しないようにし、システムにレイテンシーを追加しようとしています。この値を最大 10 まで増やすと、キーの期限切れに費やす労力を増やすことができます。トレードオフは、CPU が高くなると、潜在的にレイテンシーが高くなることです。メモリ使用率が高く、CPU 使用率の増加が許容される場合を除き、値 1 を推奨します。</p>
lazyfree-lazy-user-del (added in 6.0)	デフォルト: いいえ タイプ: 文字列 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	<p>値を yes に設定すると、DEL コマンドは UNLINK と同じように動作します。</p>

Redis 6.x で削除されたパラメータは次のとおりです。

名前	詳細	説明
lua-repl icate-comm ands	許可される値: はい/いいえ デフォルト: はい タイプ: ブール値 変更可能: はい 変更の適用: 即時	Lua 効果レプリケーションを常に有効にする が、Lua スクリプトでは有効にしません

Redis 5.0.3 パラメータの変更

パラメータグループファミリー: redis5.0

Redis 5.0 のデフォルトのパラメータグループ

- `default.redis5.0` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが無効) クラスターおよびレプリケーショングループに使用します。
- `default.redis5.0.cluster.on` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが有効) クラスターおよびレプリケーショングループに使用します。

Redis 5.0.3 で追加されたパラメータ

名前	詳細	説明
rename-co mmands	デフォルト: なし タイプ: 文字列 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	名前が変更された Redis コマンドのスペースで区切られたリスト。以下に示すのは、名前変更 に使用できるコマンドのリストの一部です。 APPEND AUTH BITCOUNT BITFIELD BITOP BITPOS BLPOP BRPOP BR POPLUSH BZPOPMIN BZPOPMAX CLIENT CLUSTER COMMAND DBSIZE DECR DECRBY DEL DISCARD DUMP ECHO EVAL

名前	詳細	説明
		EVALSHA EXEC EXISTS EXPIRE EXPIREAT FLUSHALL FLUSHDB GEOADD GEOHASH GEOPOS GEODIST GEORADIUS GEORADIUSBYMEMBER GET GETBIT GETRANGE GETSET HDEL HEXISTS HGET HGETALL HINCRBY HINCRBYFL OAT HKEYS HLEN HMGET HMSET HSET HSETNX HSTRLEN HVALS INCR INCRBY INCRBYFLOAT INFO KEYS LASTSAVE LINDEX LINSERT LLEN LPOP LPU SH LPUSHX LRANGE LREM LSET LTRIM MEMORY MGET MONITOR MOVE MSET MSETNX MULTI OBJECT PERSIST PEXPIRE PEXPIREAT PFADD PFCOUNT PFMERGE PING PSETEX PSUBSCRIBE PUBSUB PTTL PUBLISH PUNSUBSCRIBE RANDOMKEY READONLY READWRITE RENAME RENAMENX RESTORE ROLE RPOP RPOPLPUSH RPUSH RPUSHX SADD SCARD SCRIPT SDIFF SDIFFSTORE SELECT SET SETBIT SETEX SETNX SETRANGE SINTER SINTERSTORE SISMEMBER SLOWLOG SMEMBERS SMOVE SORT SPOP SRANDMEMBER SREM STRLEN SUBSCRIBE UNION UNIONSTORE SWAPDB TIME TOUCH TTL TYPE UNSUBSCRIBE UNLINK UNWATCH WAIT WATCH ZADD ZCARD ZCOUNT ZINCRBY ZINTERSTO RE ZLEXCOUNT ZPOPMAX ZPOPMIN ZRANGE ZRANGEBYLEX ZREVRANGE BYLEX ZRANGEBYSCORE ZRANK ZREM ZREMRANGEBYLEX ZREMRANGEBYRANK ZREMRANGEBYSCORE ZREVRANGE ZREVRANGEBYSCORE ZREVRANK ZSCORE

名前	詳細	説明
		ZUNIONSTORE SCAN SSCAN HSCAN ZSCAN XINFO XADD XTRIM XDEL XRA NGE XREVRANGE XLEN XREAD XGROUP XREADGROUP XACK XCLAIM XPENDING GEORADIUS_RO GEORADIUSBYMEMBER_ RO LOLWUT XSETID SUBSTR

詳細については、「[ElastiCache for Redis バージョン 5.0.6 \(拡張\)](#)」を参照してください。

Redis 5.0.0 パラメータの変更

パラメータグループファミリー: redis5.0

Redis 5.0 のデフォルトのパラメータグループ

- `default.redis5.0` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが無効) クラスターおよびレプリケーショングループに使用します。
- `default.redis5.0.cluster.on` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが有効) クラスターおよびレプリケーショングループに使用します。

Redis 5.0 で追加されたパラメータ

名前	詳細	説明
<code>stream-node-max-bytes</code>	許可される値: 0+ デフォルト: 4096 タイプ: 整数 変更可能: はい 変更の適用: 即時	ストリームデータ構造は、内部の複数のアイテムをエンコードするノードの基数ツリーです。基数ツリーの単一ノードの最大サイズをバイト単位で指定するには、この設定を使用します。0 に設定されている場合、ツリーノードのサイズは無制限です。

名前	詳細	説明
stream-node-max-entries	許可される値: 0+ デフォルト: 100 タイプ: 整数 変更可能: はい 変更の適用: 即時	ストリームデータ構造は、内部の複数のアイテムをエンコードするノードの基数ツリーです。新しいストリームエントリを追加するとき、新しいノードに切り替える前に単一ノードに含めることができるアイテムの最大数を指定するには、この設定を使用します。0 に設定されている場合、ツリーノードのアイテムの数は無制限です
active-defrag-max-scan-fields	許可される値: 1 ~ 1000000 デフォルト: 1000 タイプ: 整数 変更可能: はい 変更の適用: 即時	メインディクショナリスキャンから処理される set/hash/zset/list フィールドの最大数
lua-replicate-commands	許可される値: はい/いいえ デフォルト: はい タイプ: ブール値 変更可能: はい 変更の適用: 即時	Lua 効果レプリケーションを常に有効にするか、Lua スクリプトでは有効にしません
replica-ignore-maxmemory	デフォルト: はい タイプ: ブール値 変更可能: いいえ	プライマリから独立したアイテムを削除しないで、レプリカが maxmemory 設定を無効にするかどうかを判断します。

Redis は、コミュニティのフィードバックに応じてエンジンバージョン 5.0 でいくつかのパラメータの名前を変更しました。詳細については、「[Redis 5 の最新情報](#)」を参照してください。次の表に、新しい名前と前のバージョンとの対応を示します。

Redis 5.0 で名前が変更されたパラメータ

名前	詳細	説明
replica-lazy-flush	<p>デフォルト: はい</p> <p>タイプ: ブール値</p> <p>変更可能: いいえ</p> <p>以前の名前: slave-lazy-flush</p>	レプリカの同期中に非同期 flushDB を実行します。
client-output-buffer-limit-replica-hard-limit	<p>デフォルト: 値については、「Redis のノードタイプ固有のパラメータ」を参照してください</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>旧名: client-output-buffer-limit-slave-hard-limit</p>	Redis リードレプリカの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されます。
client-output-buffer-limit-replica-soft-limit	<p>デフォルト: 値については、「Redis のノードタイプ固有のパラメータ」を参照してください</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>旧名: client-output-buffer-limit-slave-soft-limit</p>	Redis リードレプリカの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されませんが、この条件が client-output-buffer-limit-replica-soft-seconds の間継続した場合に限ります。

名前	詳細	説明
client-output-buffer-limit-replica-soft-seconds	デフォルト: 60 タイプ: 整数 変更可能: いいえ 旧名: client-output-buffer-limit-slave-soft-seconds	Redis リードレプリカの場合: クライアントの出力バッファが、この秒数より長い時間 client-output-buffer-limit-replica-soft-limit バイトのままの場合、クライアントの接続が切断されます。
replica-allow-chaining	デフォルト: いいえ タイプ: 文字列 変更可能: いいえ 以前の名前: slave-allow-chaining	Redis のリードレプリカは自身のリードレプリカを持つことができるかどうかを決定します。
min-replicas-to-write	デフォルト: 0 タイプ: 整数 変更可能: はい 以前の名前: min-slaves-to-write 変更の適用: 即時	プライマリノードがクライアントからの書き込みを受け入れるために、使用可能でなければならないリードレプリカの数。使用可能なレプリカの数がこの数を下回った場合、プライマリノードは書き込みリクエストを受け入れなくなります。 このパラメータまたは が 0 min-replicas-max-lag の場合、プライマリノードは、使用可能なレプリカがない場合でも、常に書き込みリクエストを受け入れます。

名前	詳細	説明
min-replicas-max-lag	<p>デフォルト: 10</p> <p>タイプ: 整数</p> <p>変更可能: はい</p> <p>以前の名前: min-slaves-max-lag</p> <p>変更の適用: 即時</p>	<p>プライマリノードからリードレプリカから ping リクエストを受け取る必要がある秒数。この時間が経過してもプライマリが ping を受け取らない場合、レプリカは使用可能と見なされなくなります。使用可能なレプリカの数を下回ると min-replicas-to-write、プライマリはその時点で書き込みの受け入れを停止します。</p> <p>このパラメータまたはのいずれか min-replicas-to-write が 0 の場合、プライマリノードは使用可能なレプリカがない場合でも、常に書き込みリクエストを受け入れます。</p>
close-on-replica-write	<p>デフォルト: はい</p> <p>タイプ: ブール値</p> <p>変更可能: はい</p> <p>以前の名前: close-on-slave-write</p> <p>変更の適用: 即時</p>	<p>有効にした場合、読み取り専用レプリカに書き込もうとするクライアントの接続は切断されます。</p>

Redis 5.0 で削除されたパラメータ

名前	詳細	説明
repl-timeout	<p>デフォルト: 60</p> <p>変更可能: いいえ</p>	<p>パラメータはこのバージョンでは使用できません。</p>

Redis 4.0.10 パラメータの変更

パラメータグループファミリ: redis4.0

Redis 4.0.x のデフォルトのパラメータグループ

- `default.redis4.0` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが無効) クラスターおよびレプリケーショングループに使用します。
- `default.redis4.0.cluster.on` – このパラメータグループ、またはそこから派生したグループを、Redis (クラスターモードが有効) クラスターおよびレプリケーショングループに使用します。

Redis 4.0.10 で変更されたパラメータ

名前	詳細	説明
<code>maxmemory-policy</code>	<p>許可される値: <code>allkeys-lru</code>、<code>volatile-lru</code>、<code>allkeys-lfu</code>、<code>volatile-lfu</code>、<code>allkeys-random</code>、<code>volatile-random</code>、<code>volatile-ttl</code>、<code>noeviction</code></p> <p>デフォルト: <code>volatile-lru</code></p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の反映: 即時</p>	<p><code>maxmemory-policy</code> がバージョン 2.6.13 で追加されました。バージョン 4.0.10 では、2つの新しい許容値が追加されました。<code>allkeys-lfu</code> は、近似 LFU を使用して、すべてのキーを削除します。<code>volatile-lfu</code> は、近似 LFU を使用して、有効期限が設定されたキーを削除します。バージョン 6.2 では、データ階層化で使用するために <code>r6gd</code> ノードファミリーが導入された場合、<code>noeviction</code>、<code>volatile-lru</code> および <code>allkeys-lru</code> <code>max-memory</code> ポリシーのみが <code>r6gd</code> ノードタイプでサポートされます。</p>

Redis 4.0.10 で追加されたパラメータ

名前	詳細	説明
非同期削除パラメータ		
<code>lazyfree-lazy-eviction</code>	許可される値: はい/いいえ	削除で、非同期削除を実行します。

名前	詳細	説明
	デフォルト: いいえ	
	タイプ: ブール値	
	変更可能: はい	
	変更の反映: 即時	
lazyfree-lazy-expire	許可される値: はい/いいえ	期限切れのキーで、非同期削除を実行します。
	デフォルト: いいえ	
	タイプ: ブール値	
	変更可能: はい	
	変更の反映: 即時	
lazyfree-lazy-server-del	許可される値: はい/いいえ	値を更新するコマンドに対して非同期削除を実行します。
	デフォルト: いいえ	
	タイプ: ブール値	
	変更可能: はい	
	変更の反映: 即時	
slave-lazy-flush	許可される値: 該当なし	スレーブの同期中に非同期 flushDB を実行します。
	デフォルト: いいえ	
	タイプ: ブール値	
	変更可能: いいえ	
	変更の反映: 該当なし	

LFU パラメータ

名前	詳細	説明
lfu-log-factor	許可される値: 任意の整数 > 0 デフォルト: 10 タイプ: 整数 変更可能: はい 変更の反映: 即時	キーカウンターを飽和させるキーヒット数を決定するログ要素を設定します。
lfu-decay-time	許可される値: 任意の整数 デフォルト: 1 タイプ: 整数 変更可能: はい 変更の反映: 即時	キーカウンターをデクリメントする期間 (分単位)。
アクティブなデフラグメンテーションのパラメータ		
activedefrag	許可される値: はい/いいえ デフォルト: いいえ タイプ: ブール値 変更可能: はい 変更の反映: 即時	有効化されているアクティブなデフラグメンテーション。

名前	詳細	説明
active-defrag-ignore-bytes	許可される値: 10485760 ~ 104857600 デフォルト: 104857600 タイプ: 整数 変更可能: はい 変更の反映: 即時	アクティブなデフラグを開始するためのフラグメントの最小量。
active-defrag-threshold-lower	許可される値: 1 ~ 100 デフォルト: 10 タイプ: 整数 変更可能: はい 変更の反映: 即時	アクティブなデフラグを開始するためのフラグメントの割合。
active-defrag-threshold-upper	許可される値: 1 ~ 100 デフォルト: 100 タイプ: 整数 変更可能: はい 変更の反映: 即時	最大の労力を使用するフラグメントの最大割合。

名前	詳細	説明
active-defrag-cycle-min	許可される値: 1 ~ 75 デフォルト: 25 タイプ: 整数 変更可能: はい 変更の反映: 即時	デフラグの最小の労力 (CPU 使用率)。
active-defrag-cycle-max	許可される値: 1 ~ 75 デフォルト: 75 タイプ: 整数 変更可能: はい 変更の反映: 即時	デフラグの最大の労力 (CPU 使用率)。
クライアント出力バッファのパラメータ		
client-query-buffer-limit	許可される値: 1048576 ~ 1073741824 デフォルト: 1073741824 タイプ: 整数 変更可能: はい 変更の反映: 即時	単一のクライアントクエリバッファの最大サイズ。

名前	詳細	説明
proto-max-bulk-len	許可される値: 1048576 ~ 536870912 デフォルト: 536870912 タイプ: 整数 変更可能: はい 変更の反映: 即時	1つの要素リクエストの最大サイズ。

Redis 3.2.10 パラメータの変更

パラメータグループファミリー: redis3.2

ElastiCache for Redis 3.2.10 では、追加のパラメータはサポートされていません。

Redis 3.2.6 パラメータの変更

パラメータグループファミリー: redis3.2

Redis 3.2.6 では、追加でサポートされているパラメータはありません。

Redis 3.2.4 パラメータの変更

パラメータグループファミリー: redis3.2

Redis 3.2.4 から、2つのデフォルトのパラメータグループがあります。

- `default.redis3.2` – Redis 3.2.4 を実行する場合は、Redis (クラスターモードが無効) レプリケーショングループを作成し、Redis 3.2.4 のその他の機能を引き続き使用する場合は、このパラメータグループまたはそこから派生したパラメータグループを指定します。
- `default.redis3.2.cluster.on` – Redis (クラスターモードが有効) レプリケーショングループを作成する場合は、このパラメータグループまたはそこから派生したパラメータグループを指定します。

トピック

- [Redis 3.2.4 の新しいパラメータ](#)

• [Redis 3.2.4 \(拡張\) で変更されたパラメータ](#)

Redis 3.2.4 の新しいパラメータ

パラメータグループファミリー: redis3.2

Redis 3.2.4 では、次のパラメータが追加でサポートされます。

名前	詳細	説明
list-max-ziplist-size	デフォルト: -2 タイプ: 整数 変更可能: いいえ	<p>リストは、領域を節約する特殊な方法でエンコードされます。内部リストノードあたり許可されるエントリの数は、要素の固定最大サイズまたは最大数として指定できます。最大固定サイズには、-5~-1 を使用します。この意味は次のとおりです。</p> <ul style="list-style-type: none"> • -5: 最大サイズ: 64 KB - 通常のワークロードには推奨されません • -4: 最大サイズ: 32 KB - 推奨されません • -3: 最大サイズ: 16 KB - 推奨されません • -2: 最大サイズ: 8 KB - 推奨 • -1: 最大サイズ: 4 KB - 推奨 • 正の値は、リストノードあたり、最大でその数の要素まで保存することを意味します。
list-compress-depth	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>リストは、圧縮される場合もあります。圧縮の深さは、圧縮から除外するリストの端からのクイックリスト ziplist ノードの数です。リストの先頭と末尾は、プッシュおよびポップオペレーションを高速にするために常に圧縮されません。設定は以下のとおりです。</p>

名前	詳細	説明
		<ul style="list-style-type: none">0: すべての圧縮を無効にします。1: 先頭から末尾までの最初のノードで圧縮を開始します。 先頭->ノード->ノード->...->ノード->末尾 先頭と末尾を除くすべてのノードで圧縮を実行します。2: 先頭から末尾までの2番目のノードで圧縮を開始します。 先頭->次->ノード->ノード->...->ノード->前->末尾 先頭、次、前、末尾は圧縮されません。他のすべてのノードで圧縮を実行します。その他

名前	詳細	説明
cluster-enabled	デフォルト: no/yes * タイプ: 文字列 変更可能: いいえ	<p>これがクラスターモードの Redis (クラスターモードが有効) レプリケーショングループである (yes) か、非クラスターモードの Redis (クラスターモードが有効) レプリケーショングループである (no) かを示します。クラスターモードの Redis (クラスターモードが有効) レプリケーショングループは、最大 500 のノードグループにデータを分割できます。</p> <p>* Redis 3.2.x には 2 つのデフォルトのパラメータグループがあります。</p> <ul style="list-style-type: none">• default.redis3.2 - デフォルト値: no。• default.redis3.2.cluster.on - デフォルト値: yes。

名前	詳細	説明
cluster-require-full-coverage	デフォルト: いいえ タイプ: ブール値 変更可能: はい 変更の適用: 即時	<p>yes に設定された場合、いずれのノードの処理対象になっていないハッシュスロットが検出された場合、クラスターモードの edis (クラスターモードが有効) ノードがクエリの受け入れを停止します。このように、クラスターが部分的にダウンしている場合、クラスターは使用できなくなります。すべてのスロットが再び処理対象になると、クラスターは自動的に再び使用可能になります。</p> <p>ただし、まだ処理対象になっているキー空間の部分に対するクエリを受け入れ続けるようにクラスターのサブセットが機能していることが必要な場合があります。その場合は、cluster-require-full-coverage オプションを no に設定するだけです。</p>

名前	詳細	説明
hll-spars e-max-byt es	デフォルト: 3000 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>HyperLogLog スパース表現のバイト数制限。この制限には 16 バイトのヘッダーが含まれます。スパース表現 HyperLogLog を使用するがこの制限を超えると、高密度表現に変換されます。</p> <p>16,000 より大きい値はお勧めしません。その時点では、デンスな表現の方がメモリ効率が高くなるためです。</p> <p>PFADD の速度を下げすぎることなく領域効率の良いエンコードの利点を活かせる (スパースなエンコードで $O(N)$ になる) ように、値は約 3,000 にすることをお勧めします。CPU が問題ではないがスペースが問題で、データセットがカーディナリティが 0~15000 の範囲 HyperLogLogs の多くので構成されている場合、値は 10000 まで引き上げることができません。</p>

名前	詳細	説明
reserved-memory-percent	デフォルト: 25 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>非データ用に確保されているノードのメモリの割合。デフォルトでは、ノードのメモリがすべて消費されるまで Redis データフットプリントは増加します。この場合、メモリページングが大量に行われるため、ノードパフォーマンスが低下する可能性が高くなります。使用可能なメモリの一部を Redis 以外の用途に確保しておくことで、ページングの量を減らすことができます。</p> <p>このパラメータは に固有であり ElastiCache、標準の Redis ディストリビューションの一部ではありません。</p> <p>詳細については、「reserved-memory 」および「予約メモリの管理」を参照してください。</p>

Redis 3.2.4 (拡張) で変更されたパラメータ

パラメータグループファミリー: redis3.2

Redis 3.2.4 では、以下のパラメータが変更されました。

名前	詳細	変更
activeresharding	変更可能: パラメータグループがいずれのキャッシュクラスターにも関連付けられていない場合は、はい。それ以外の場合は No です。	変更可能は No でした。
databases	変更可能: パラメータグループがいずれのキャッシュクラスターにも関連付けられていな	変更可能は No でした。

名前	詳細	変更
	い場合は、はい。それ以外の場合は No です。	
appendonly	デフォルト: オフ 変更可能: いいえ	以前のバージョンの Redis からアップグレードする場合は、最初に appendonly をオフにする必要があります。
appendfsync	デフォルト: オフ 変更可能: いいえ	以前のバージョンの Redis からアップグレードする場合は、最初に appendfsync をオフにする必要があります。
repl-timeout	デフォルト: 60 変更可能: いいえ	現在はデフォルト値 60 で、変更できません。
tcp-keepalive	デフォルト: 300	デフォルト値は 0 でした。
list-max-ziplist-entries		パラメータは使用できなくなりました。
list-max-ziplist-value		パラメータは使用できなくなりました。

Redis 2.8.24 (拡張) で追加されたパラメータ

パラメータグループファミリー: redis2.8

Redis 2.8.24 では、追加でサポートされているパラメータはありません。

Redis 2.8.23 (拡張) で追加されたパラメータ

パラメータグループファミリー: redis2.8

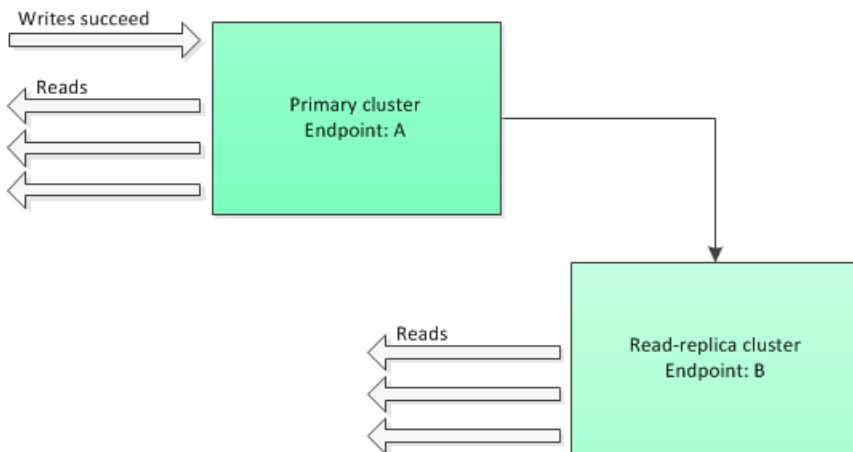
Redis 2.8.23 では、以下のパラメータが追加でサポートされます。

名前	詳細	説明
close-on-slave-write	デフォルト: はい タイプ: 文字列 (はい/いいえ) 変更可能: はい 変更の適用: 即時	有効にした場合、読み取り専用レプリカに書き込もうとするクライアントの接続は切断されます。

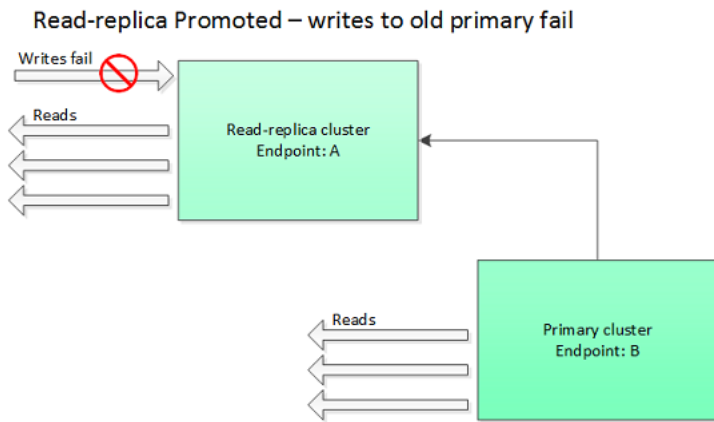
close-on-slave-write の仕組み

close-on-slave-write パラメータは Amazon によって導入され ElastiCache、リードレプリカをプライマリーに昇格させるため、プライマリノードとリードレプリカノードがロールをスワップしたときのクラスターの応答をより詳細に制御できます。

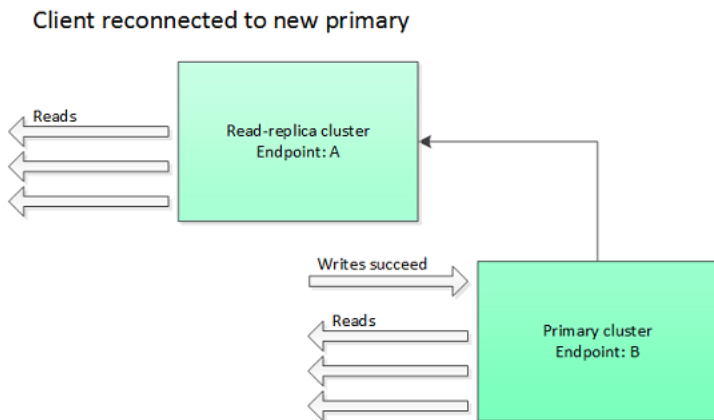
Before read-replica promotion



リードレプリカクラスターが、マルチ AZ 対応レプリケーショングループのフェイルオーバー以外の理由で、プライマリーに昇格する場合、クライアントは引き続きエンドポイント A に書き込もうとします。エンドポイント A はこの時点でリードレプリカのエンドポイントであるため、これらの書き込みは失敗します。これは、ElastiCache を導入する前の Redis の動作 close-on-replica-write であり、を無効にする場合の動作です close-on-replica-write。



`close-on-replica-write` が有効になっていると、クライアントがリードレプリカに書き込もうとするたびに、クラスターへのクライアントの接続は切断されます。アプリケーションロジックは、切断を検出し、DNS テーブルを確認して、プライマリエンドポイント (この時点でエンドポイント B になっている) に再接続する必要があります。



を無効にする場合 `close-on-replica-write`

`close-on-replica-write` を無効にすると、障害が発生しているクラスターに書き込まれることとなります。それでは、なぜ `close-on-replica-write` を無効にするのでしょうか。

前述したように、`close-on-replica-write` が有効になっていると、クライアントがリードレプリカに書き込もうとするたびに、クラスターへのクライアントの接続は切断されます。ノードへの新しい接続の確立には時間がかかります。したがって、レプリカへの書き込みリクエストの結果として切断および再接続が行われると、同じ接続を介して提供される読み取りリクエストのレイテンシーにも影響します。この効果は、新しい接続が確立されるまで維持されます。特に、読み取りが多いアプリケーションや、レイテンシーの影響を受けやすいアプリケーションの場合、読み取りパフォーマンスが下がらないように、クライアントを接続したままにすることができます。

Redis 2.8.22 (拡張) で追加されたパラメータ

パラメータグループファミリー: redis2.8

Redis 2.8.22 では、追加でサポートされているパラメータはありません。

Important

- Redis バージョン 2.8.22 から、プライマリクラスターとレプリカクラスターに `repl-backlog-size` が適用されるようになりました。
- Redis バージョン 2.8.22 以降では、`repl-timeout` パラメータはサポートされていません。変更した場合、と同様にデフォルト (60 秒) で上書き ElastiCache されません `appendonly`。

次のパラメータはサポートされなくなりました。

- `appendonly`
- `appendfsync`
- `repl-timeout`

Redis 2.8.21 で追加されたパラメータ

パラメータグループファミリー: redis2.8

Redis 2.8.21 では、追加でサポートされているパラメータはありません。

Redis 2.8.19 で追加されたパラメータ

パラメータグループファミリー: redis2.8

Redis 2.8.19 では、追加でサポートされているパラメータはありません。

Redis 2.8.6 で追加されたパラメータ


パラメータグループファミリー: redis2.8

Redis 2.8.6 では、次のパラメータが追加でサポートされます。

名前	詳細	説明
min-slaves-max-lag	デフォルト: 10 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>プライマリノードからリードレプリカから ping リクエストを受け取る必要がある秒数。この時間が経過してもプライマリが ping を受け取らない場合、レプリカは使用可能と見なされなくなります。使用可能なレプリカの数 が を下回ると min-slaves-to-write、プライマリはその時点で書き込みの受け入れを停止します。</p> <p>このパラメータまたは min-slaves-to-write が 0 の場合、プライマリノードは、使用可能なレプリカがない場合でも、常に書き込みリクエストを受け入れます。</p>
min-slaves-to-write	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>プライマリノードがクライアントからの書き込みを受け入れるために、使用可能でなければならないリードレプリカの数。使用可能なレプリカの数がこの数を下回った場合、プライマリノードは書き込みリクエストを受け入れなくなります。</p> <p>このパラメータまたは min-slaves-max-lag が 0 の場合、プライマリノードは、使用可能なレプリカがない場合でも、常に書き込みリクエストを受け入れます。</p>
notify-keyspace-events	デフォルト: (空の文字列) タイプ: 文字列	Redis はクライアントに通知できる keyspace のタイプ。各イベントタイプは 1 文字で表されます。

名前	詳細	説明
	変更可能: はい 変更の適用: 即時	<ul style="list-style-type: none"> • K — Keyspace イベント、プレフィックス <code>__keyspace@<db>__</code> を付けて発行 • E — Key-event イベント、プレフィックス <code>__keyevent@<db>__</code> を付けて発行 • g — 固有でない汎用コマンド (DEL、EXPIRE、RENAME など) • \$ — 文字列コマンド • l — リストコマンド • s — 設定コマンド • h — ハッシュコマンド • z — ソート対象セットコマンド • x — 期限切れのイベント (キーの期限が切れるたびにイベントが生成されます) • e — 削除されたイベント (maxmemory に達したためにキーが削除された場合にイベントが生成されます) • A — <code>g\$lshzxe</code> のエイリアス

名前	詳細	説明
		<p>これらのイベントタイプは自由に組み合わせることができます。たとえば、AKE は Redis がすべてのイベントタイプの通知を発行できることを意味します。</p> <p>上に挙げられた文字以外の文字を使用しないでください。使用しようとする、エラーメッセージが表示されます。</p> <p>デフォルトでは、このパラメータは空の文字列に設定されます。これは、keyspace イベント通知が無効であることを意味します。</p>

名前	詳細	説明
repl-backlog-size	デフォルト: 1048576 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>プライマリノードバックログバッファのサイズ (バイト単位)。バックログは、プライマリノードのデータの更新を記録するために使用されます。リードレプリカは、プライマリに接続すると、部分同期 (psync) の実行を試みます。このとき、プライマリノードに追いつくことができるようにバックログからデータを適用します。psync に失敗した場合は、完全同期が必要です。</p> <p>このパラメータの最小値は 16384 です。</p> <div data-bbox="1008 957 1507 1266"><p> Note</p><p>Redis 2.8.22 から、このパラメータはプライマリクラスターとリードレプリカに適用されます。</p></div>


名前	詳細	説明
repl-backlog-ttl	デフォルト: 3600 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>プライマリノードがバックログバッファを保持する秒数。最後のレプリカノードが切断されたときから、バックログ内のデータは repl-backlog-ttl の期限が切れるまで変更されません。レプリカがこの時間内にプライマリに接続されない場合、プライマリはバックログバッファを解放しません。レプリカが最終的に再接続した場合、プライマリとの完全同期を実行する必要があります。</p> <p>このパラメータを 0 に設定した場合、バックログバッファは解放されません。</p>
repl-timeout	デフォルト: 60 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>次のタイムアウト時間 (秒単位) を表します。</p> <ul style="list-style-type: none"> • 同期中の一括データ転送 (リードレプリカの観点から) • プライマリノードのタイムアウト (レプリカの観点から) • レプリカのタイムアウト (プライマリノードの観点から)

Redis 2.6.13 パラメータ

パラメータグループファミリー: redis2.6

Redis 2.6.13 は、でサポートされている Redis の最初のバージョンでした ElastiCache。次の表は、が ElastiCache サポートする Redis 2.6.13 パラメータを示しています。

名前	詳細	説明
activereshashing	デフォルト: はい タイプ: 文字列 (はい/いいえ) 変更可能: はい 変更の適用: 作成時	<p>Redis のアクティブな再ハッシュ機能を有効にするかどうかを決定します。主要なハッシュテーブルは、1 秒あたり 10 回再ハッシュされます。再ハッシュ操作ごとに 1 ミリ秒の CPU が消費されます。</p> <p>パラメータグループを作成するとき、この値を設定します。クラスターに新しいパラメータグループを割り当てるとき、この値は以前のパラメータグループと新しいパラメータグループで一致している必要があります。</p>
appendonly	デフォルト: いいえ タイプ: 文字列 変更可能: はい 変更の適用: 即時	<p>Redis の AOF (Append Only File) 機能を有効または無効にします。AOF は、キャッシュ内のデータを変更する Redis コマンドをキャプチャし、特定のノード障害からの復元に使用されます。</p> <p>デフォルト値は no です (AOF が無効であることを意味します)。AOF を有効にするには、このパラメータを yes に設定します。</p> <p>詳細については、「障害の軽減」を参照してください。</p> <div data-bbox="829 1367 1507 1724"><p>Note</p><p>AOF (Append Only File) は、cache.t1.micro ノードおよび cache.t2.* ノードではサポートされません。このタイプのノードの場合、appendonly パラメータ値は無視されます。</p></div>

名前	詳細	説明
		<p> Note</p> <p>マルチ AZ レプリケーショングループでは、AOF は許可されません。</p>
appendfsync	<p>デフォルト: everysec</p> <p>タイプ: 文字列</p> <p>変更可能: はい</p> <p>変更の適用: 即時</p>	<p>appendonly を [yes] に設定すると、AOF 出力バッファをディスクに書き込む頻度が制御されます。</p> <ul style="list-style-type: none"> no — バッファは必要に応じてディスクにフラッシュされます。 everysec — バッファは 1 秒に 1 回フラッシュされます。これがデフォルトです。 always — バッファは、クラスターが変更されるたびにフラッシュされます。 Appendfsync は、バージョン 2.8.22 以降ではサポートされていません。
client-output-buffer-limit-normal-hard-limit	<p>デフォルト: 0</p> <p>タイプ: 整数</p> <p>変更可能: はい</p> <p>変更の適用: 即時</p>	<p>クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されます。デフォルトは 0 です (ハード制限なし)。</p>
client-output-buffer-limit-normal-soft-limit	<p>デフォルト: 0</p> <p>タイプ: 整数</p> <p>変更可能: はい</p> <p>変更の適用: 即時</p>	<p>クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-normal-soft-seconds の間継続した場合に限ります。デフォルトは 0 です (ソフト制限なし)。</p>

名前	詳細	説明
client-output-buffer-limit-normal-soft-seconds	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	クライアントの出力バッファが、この秒数より長い時間 client-output-buffer-limit-normal-soft-limit バイトのままの場合、クライアントの接続が切断されます。デフォルトは 0 です (時間制限なし)。
client-output-buffer-limit-pubsub-hard-limit	デフォルト: 33554432 タイプ: 整数 変更可能: はい 変更の適用: 即時	Redis 発行/サブスクライブクライアントの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されます。
client-output-buffer-limit-pubsub-soft-limit	デフォルト: 8388608 タイプ: 整数 変更可能: はい 変更の適用: 即時	Redis 発行/サブスクライブクライアントの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-pubsub-soft-seconds の間継続した場合に限ります。
client-output-buffer-limit-pubsub-soft-seconds	デフォルト: 60 タイプ: 整数 変更可能: はい 変更の適用: 即時	Redis 発行/サブスクライブクライアントの場合: クライアントの出力バッファがこの秒数より長い間 client-output-buffer-limit-pubsub-soft-limit バイトのままの場合、クライアントの接続が切断されます。
client-output-buffer-limit-slave-hard-limit	デフォルト: 値については、 「Redis のノードタイプ固有のパラメータ」 を参照してください タイプ: 整数 変更可能: いいえ	Redis リードレプリカの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されます。

名前	詳細	説明
client-output-buffer-limit-slave-soft-limit	<p>デフォルト: 値については、「Redis のノードタイプ固有のパラメータ」を参照してください</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p>	<p>Redis リードレプリカの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-slave-soft-seconds の間継続した場合に限ります。</p>
client-output-buffer-limit-slave-soft-seconds	<p>デフォルト: 60</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p>	<p>Redis リードレプリカの場合: クライアントの出力バッファが、この秒数より長い時間 client-output-buffer-limit-slave-soft-limit バイトのままの場合、クライアントの接続が切断されます。</p>
databases	<p>デフォルト: 16</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>変更の適用: 作成時</p>	<p>論理パーティションデータベース数は分割されます。この値を低く抑えることをお勧めします。</p> <p>パラメータグループを作成するとき、この値を設定します。クラスターに新しいパラメータグループを割り当てるとき、この値は以前のパラメータグループと新しいパラメータグループで一致している必要があります。</p>
hash-max-ziplist-entries	<p>デフォルト: 512</p> <p>タイプ: 整数</p> <p>変更可能: はい</p> <p>変更の適用: 即時</p>	<p>ハッシュに使用されるメモリ量を決定します。エントリが指定された数より少ないハッシュは、領域を節約する特殊なエンコードを使用して格納されます。</p>

名前	詳細	説明
hash-max-ziplist-value	デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: 即時	ハッシュに使用されるメモリ量を決定します。エントリが指定されたバイト数より小さいハッシュは、領域を節約する特殊なエンコードを使用して格納されます。
list-max-ziplist-entries	デフォルト: 512 タイプ: 整数 変更可能: はい 変更の適用: 即時	リストに使用されるメモリ量を決定します。エントリが指定された数より少ないリストは、領域を節約する特殊なエンコードを使用して格納されます。
list-max-ziplist-value	デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: 即時	リストに使用されるメモリ量を決定します。エントリが指定されたバイト数より小さいリストは、領域を節約する特殊なエンコードを使用して格納されます。
lua-time-limit	デフォルト: 5000 タイプ: 整数 変更可能: いいえ	<p> がスクリプトを停止する ElastiCache アクションを実行するまでの、Lua スクリプトの最大実行時間をミリ秒単位で指定します。 </p> <p> lua-time-limit を超過した場合、すべての Redis コマンドによりエラーが <code>___-BUSY</code> の形式で返されます。この状態は多くの重要な Redis オペレーションに干渉する可能性があるため、まず SCRIPT KILL コマンドを発行 ElastiCache します。これが失敗した場合、Redis は強制的に再起動 ElastiCache します。 </p>

名前	詳細	説明
maxclients	<p>デフォルト: 65000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t2.medium デフォルト: 20000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t2.small デフォルト: 20000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t2.micro デフォルト: 20000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t4g.micro デフォルト: 20000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t3.medium デフォルト: 46000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p> <p>t3.small デフォルト: 46000</p> <p>タイプ: 整数</p> <p>変更可能: いいえ</p>	一度に接続できるクライアントの最大数。

名前	詳細	説明
	t3.micro デフォルト: 20000 タイプ: 整数 変更可能: いいえ	
maxmemory-policy	デフォルト: volatile-lru タイプ: 文字列 変更可能: はい 変更の適用: 即時	メモリの最大使用量に到達したときのキーの削除ポリシー。 有効な値は次のとおりです。volatile-lru allkeys-lru volatile-random allkeys-random volatile-ttl noeviction 詳細については、「 Redis を LRU キャッシュとして使用する 」を参照してください。
maxmemory-samples	デフォルト: 3 タイプ: 整数 変更可能: はい 変更の適用: 即時	least-recently-used (LRU) および time-to-live (TTL) 計算の場合、このパラメータはチェックするキーのサンプルサイズを表します。デフォルトで、Redis は 3 個のキーを選択し、最も長い間使用されていないキーを使用します。

名前	詳細	説明
reserved-memory	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>非データの使用に確保された合計メモリ (バイト単位)。デフォルトでは、Redis ノードは、ノードの maxmemory を消費するまで大きくなります (「Redis のノードタイプ固有のパラメータ」を参照)。この場合、メモリページングが大量に行われるため、ノードパフォーマンスが低下する可能性が高くなります。使用可能なメモリの一部を Redis 以外の用途に確保しておくことで、ページングの量を減らすことができます。</p> <p>このパラメータは に固有であり ElastiCache、標準の Redis ディストリビューションの一部ではありません。</p> <p>詳細については、「reserved-memory-percent」および「予約メモリの管理」を参照してください。</p>
set-max-intset-entries	デフォルト: 512 タイプ: 整数 変更可能: はい 変更の適用: 即時	<p>特定のタイプのセットに使用されるメモリの量を決定します (64 ビット符号付き整数の範囲に収まる基数 10 の整数である文字列)。エントリが指定された数より少ないセットは、領域を節約する特殊なエンコードを使用して格納されます。</p>
slave-allow-chaining	デフォルト: いいえ タイプ: 文字列 変更可能: いいえ	<p>Redis のリードレプリカは自身のリードレプリカを持つことができるかどうかを決定します。</p>

名前	詳細	説明
slowlog-log-slower-than	デフォルト: 10000 タイプ: 整数 変更可能: はい 変更の適用: 即時	Redis の Slow Log 機能によりコマンドを記録する最大実行時間 (マイクロ秒単位)。
slowlog-max-len	デフォルト: 128 タイプ: 整数 変更可能: はい 変更の適用: 即時	Redis Slow Log の最大長。
tcp-keepalive	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	0 以外の値 (N) に設定した場合、接続が維持されていることを確認するためにノードクライアントが N 秒ごとにポーリングされます。デフォルト設定の 0 では、このようなポーリングが行われません。 <div style="border: 1px solid #f08080; border-radius: 10px; padding: 10px;"><p> Important</p><p>Redis バージョン 3.2.4 では、このパラメータのアスペクト値が一部変更されています。 Redis 3.2.4 (拡張) で変更されたパラメータ を参照してください。</p></div>


名前	詳細	説明
timeout	デフォルト: 0 タイプ: 整数 変更可能: はい 変更の適用: 即時	ノードがタイムアウトまで待機する秒数。値は次のとおりです。 <ul style="list-style-type: none"> • 0 – アイドル状態のクライアントは切断しません。 • 1-19 – 無効な値です。 • ≥ 20 – ノードがアイドル状態のクライアントを切断するまでに待機する秒数。
zset-max-ziplist-entries	デフォルト: 128 タイプ: 整数 変更可能: はい 変更の適用: 即時	ソート対象セットに使用されるメモリ量を決定します。要素が指定された数より少ないソート対象セットは、領域を節約する特殊なエンコードを使用して格納されます。
zset-max-ziplist-value	デフォルト: 64 タイプ: 整数 変更可能: はい 変更の適用: 即時	ソート対象セットに使用されるメモリ量を決定します。エントリが指定されたバイト数より小さいソート対象セットは、領域を節約する特殊なエンコードを使用して格納されます。

Note

Redis 2.6.13 クラスターにパラメータグループを指定しない場合、デフォルトのパラメータグループ (default.redis2.6) が使用されます。デフォルトのパラメータグループ内のパラメータは、どれも値を変更できません。ただし、いつでもカスタムパラメータグループを作成して、クラスターに割り当てることができます。

Redis のノードタイプ固有のパラメータ

ほとんどのパラメータの値は 1 つですが、一部のパラメータには、使用されているノードタイプによって複数の値が設定されることがあります。次の表は、各ノードタイプの maxmemory、client-output-buffer-limit-slave-hard-limit、および client-output-buffer-limit-slave-soft-limit パラメータのデフォルト値を示しています。maxmemory の値は、ノードでデータやその他の用途に使用できる最大バイト数です。詳細については、「[使用可能なメモリ](#)」を参照してください。

 Note

maxmemory パラメータは変更できません。

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.t1.micro	142606336	14260633	14260633
cache.t2.micro	581959680	58195968	58195968
cache.t2.small	1665138688	166513868	166513868
cache.t2.medium	3461349376	346134937	346134937
cache.t3.micro	536870912	53687091	53687091
cache.t3.small	1471026299	147102629	147102629
cache.t3.medium	3317862236	331786223	331786223
cache.t4g.micro	536870912	53687091	53687091
cache.t4g.small	1471026299	147102629	147102629
cache.t4g.medium	3317862236	331786223	331786223
cache.m1.small	943718400	94371840	94371840

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.m1.medium	3093299200	309329920	309329920
cache.m1.large	7025459200	702545920	702545920
cache.m1.xlarge	14889779200	1488977920	1488977920
cache.m2.xlarge	17091788800	1709178880	1709178880
cache.m2.2xlarge	35022438400	3502243840	3502243840
cache.m2.4xlarge	70883737600	7088373760	7088373760
cache.m3.medium	2988441600	309329920	309329920
cache.m3.large	6501171200	650117120	650117120
cache.m3.xlarge	14260633600	1426063360	1426063360
cache.m3.2xlarge	29989273600	2998927360	2998927360
cache.m4.large	6892593152	689259315	689259315
cache.m4.xlarge	15328501760	1532850176	1532850176
cache.m4.2xlarge	31889126359	3188912636	3188912636
cache.m4.4xlarge	65257290629	6525729063	6525729063
cache.m4.10xlarge	166047614239	16604761424	16604761424
cache.m5.large	6854542746	685454275	685454275
cache.m5.xlarge	13891921715	1389192172	1389192172
cache.m5.2xlarge	27966669210	2796666921	2796666921
cache.m5.4xlarge	56116178125	5611617812	5611617812

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.m5.12xlarge	168715971994	16871597199	16871597199
cache.m5.24xlarge	337500562842	33750056284	33750056284
cache.m6g.large	6854542746	685454275	685454275
cache.m6g.xlarge	13891921715	1389192172	1389192172
cache.m6g.2xlarge	27966669210	2796666921	2796666921
cache.m6g.4xlarge	56116178125	5611617812	5611617812
cache.m6g.8xlarge	111325552312	11132555231	11132555231
cache.m6g.12xlarge	168715971994	16871597199	16871597199
cache.m6g.16xlarge	225000375228	22500037523	22500037523
cache.c1.xlarge	6501171200	650117120	650117120
cache.r3.large	14470348800	1468006400	1468006400
cache.r3.xlarge	30513561600	3040870400	3040870400
cache.r3.2xlarge	62495129600	6081740800	6081740800
cache.r3.4xlarge	126458265600	12268339200	12268339200
cache.r3.8xlarge	254384537600	24536678400	24536678400
cache.r4.large	13201781556	1320178155	1320178155
cache.r4.xlarge	26898228839	2689822883	2689822883
cache.r4.2xlarge	54197537997	5419753799	5419753799
cache.r4.4xlarge	108858546586	10885854658	10885854658

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.r4.8xlarge	218255432090	21825543209	21825543209
cache.r4.16xlarge	437021573120	43702157312	43702157312
cache.r5.large	14037181030	1403718103	1403718103
cache.r5.xlarge	28261849702	2826184970	2826184970
cache.r5.2xlarge	56711183565	5671118356	5671118356
cache.r5.4xlarge	113609865216	11360986522	11360986522
cache.r5.12xlarge	341206346547	34120634655	34120634655
cache.r5.24xlarge	682485973811	68248597381	68248597381
cache.r6g.large	14037181030	1403718103	1403718103
cache.r6g.xlarge	28261849702	2826184970	2826184970
cache.r6g.2xlarge	56711183565	5671118356	5671118356
cache.r6g.4xlarge	113609865216	11360986522	11360986522
cache.r6g.8xlarge	225000375228	22500037523	22500037523
cache.r6g.12xlarge	341206346547	34120634655	34120634655
cache.r6g.16xlarge	450000750456	45000075046	45000075046
cache.r6gd.xlarge	28261849702	2826184970	2826184970
cache.r6gd.2xlarge	56711183565	5671118356	5671118356
cache.r6gd.4xlarge	113609865216	11360986522	11360986522
cache.r6gd.8xlarge	225000375228	22500037523	22500037523

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.r6gd.12xlarge	341206346547	34120634655	34120634655
cache.r6gd.16xlarge	450000750456	45000075046	45000075046
cache.r7g.large	14037181030	1403718103	1403718103
cache.r7g.xlarge	28261849702	2826184970	2826184970
cache.r7g.2xlarge	56711183565	5671118356	5671118356
cache.r7g.4xlarge	113609865216	11360986522	11360986522
cache.r7g.8xlarge	225000375228	22500037523	22500037523
cache.r7g.12xlarge	341206346547	34120634655	34120634655
cache.r7g.16xlarge	450000750456	45000075046	45000075046
cache.m7g.large	6854542746	685454275	685454275
cache.m7g.xlarge	13891921715	1389192172	1389192172
cache.m7g.2xlarge	27966669210	2796666921	2796666921
cache.m7g.4xlarge	56116178125	5611617812	5611617812
cache.m7g.8xlarge	111325552312	11132555231	11132555231
cache.m7g.12xlarge	168715971994	16871597199	16871597199
cache.m7g.16xlarge	225000375228	22500037523	22500037523
cache.c7gn.large	3317862236	1403718103	1403718103
cache.c7gn.xlarge	6854542746	2826184970	2826184970
cache.c7gn.2xlarge	13891921715	5671118356	5671118356

ノードの種類	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.c7gn.4xlarge	27966669210	11360986522	11360986522
cache.c7gn.8xlarge	56116178125	22500037523	22500037523
cache.c7gn.12xlarge	84357985997	34120634655	34120634655
cache.c7gn.16xlarge	113609865216	45000075046	45000075046

Note

現在の世代のインスタンスタイプはすべて、デフォルトで Amazon Virtual Private Cloud VPC で作成されます。

T1 インスタンスはマルチ AZ をサポートしません。

T1 および T2 インスタンスでは、Redis AOF をサポートしていません。

Redis 構成変数 `appendonly` および `appendfsync` Redis バージョン 2.8.22 以降ではサポートされていません。

Redis ElastiCache のスケーリング

ElastiCache サーバーレスのスケーリング

ElastiCache サーバーレスは、ワークロードトラフィックの増減に自動的に対応します。ElastiCache サーバーレスキャッシュごとに、は CPU、メモリ、ネットワークなどのリソースの使用率 ElastiCache を継続的に追跡します。これらのリソースのいずれかに制約がある場合、ElastiCache Serverless は新しいシャードを追加し、アプリケーションをダウンタイムなく新しいシャードにデータを再分散することでスケールアウトします。キャッシュデータストレージの BytesUsedForCache メトリクスとコンピューティング使用量の ElastiCacheProcessingUnits (ECPU) をモニタリング CloudWatch することで、のキャッシュで消費されているリソースをモニタリングできます。

スケーリング制限を設定してコストを管理する

キャッシュコストを抑えるために、キャッシュデータストレージとキャッシュの ECPU/秒の両方に使用量の上限を設定することができます。そうすることで、キャッシュ使用量が設定した上限を超えることがなくなります。

スケーリングの上限を設定した場合、キャッシュが上限に達すると、アプリケーションのキャッシュパフォーマンスが低下する可能性があります。キャッシュデータストレージの最大数を設定し、キャッシュデータストレージが最大数に達すると、は LRU ロジックを使用して、有効期限 (TTL) が設定されているキャッシュ内のデータの削除 ElastiCache を開始します。削除できるデータがない場合、追加データを書き込むリクエストにはメモリ不足(OOM) エラーメッセージが表示されます。ECPU/秒の最大値を設定し、ワークロードのコンピューティング使用率がこの値を超えると、ElastiCache は Redis リクエストのロットリングを開始します。

BytesUsedForCache または に上限を設定する場合はElastiCacheProcessingUnits、キャッシュがこれらの制限に近づいたときに通知されるように、上限よりも低い値でCloudWatch アラームを設定することを強くお勧めします。設定した上限の 75% にアラームを設定することをお勧めします。CloudWatch アラームの設定方法については、「ドキュメント」を参照してください。

ElastiCache Serverless による事前スケーリング

ElastiCache サーバーレスの事前スケーリング

事前スケーリングは事前ウォーミングとも呼ばれ、ElastiCache キャッシュでサポートされる最小制限を設定できます。これらの最小値は、1 秒あたりの ElastiCache 処理単位 (ECPUs またはデータストレージ) に設定できます。これは、予想されるスケーリングイベントの準備に役立ちます。例えば、ゲーム会社が新しいゲームのローンチから最初の 1 分以内にログインが 5 倍増加すると想定している場合、この大幅な使用量の急増に対してキャッシュの準備をすることができます。

ElastiCache コンソール、CLI、または API を使用して事前スケーリングを実行できます。

ElastiCache Serverless は 60 分以内にキャッシュで使用可能な ECPUs/秒を更新し、最小制限の更新が完了するとイベント通知を送信します。

事前スケーリングの仕組み

ECPU /秒またはデータストレージの最小制限がコンソール、CLI、または API を介して更新されると、その新しい制限は 1 時間以内に利用できます。ElastiCache Serverless は空のキャッシュで 30K ECPUs/秒をサポートし、レプリカからの読み取り機能を使用すると最大 90K ECPUs /秒をサポートします。10 ~ 12 分ごとに ECPUs/秒 ElastiCache を使用できます。このスケーリング速度

は、ほとんどのワークロードで十分です。今後のスケーリングイベントがこのレートを超えることが予想される場合は、ピークイベントの少なくとも 60 分前に、最小 ECPUs/秒をピーク ECPUs/秒に設定することをお勧めします。そうしないと、アプリケーションのレイテンシーが増加し、リクエストのスロットリングが発生する可能性があります。

最小制限の更新が完了すると、ElastiCache Serverless は 1 秒あたりの新しい最小 ECPUs または新しい最小ストレージの計測を開始します。これは、アプリケーションがキャッシュでリクエストを実行していない場合や、データストレージの使用量が最小値を下回っている場合にも発生します。現在の設定から最小制限を引き下げると、更新は即時に行われるため、ElastiCache サーバーレスは新しい最小制限ですぐに計測を開始します。

Note

- 最小使用制限を設定すると、実際の使用量が最小使用制限を下回っていても、その制限に対して課金されます。最小使用制限を超える ECPU またはデータストレージの使用量には、通常料金が課金されます。例えば、最小使用制限を 100,000 ECPUs/秒に設定すると、使用量がその最小設定よりも低い場合でも、1 時間あたり 1.224 USD 以上 (us-east-1 の ECPU 料金を使用) が課金されます。
- ElastiCache Serverless は、キャッシュ上の集約レベルでリクエストされた最小スケールをサポートします。ElastiCache Serverless は、スロットあたり最大 30K ECPUs/秒 (READONLY 接続を使用してレプリカから読み取るを使用する場合、90K ECPUs/秒) もサポートします。ベストプラクティスとして、アプリケーションは Redis スロット間のキー分散とキー間のトラフィックが可能な限り均一であることを確認する必要があります。

コンソールと を使用したスケーリング制限の設定 AWS CLI

AWS コンソールを使用したスケーリング制限の設定

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、変更対象のキャッシュで実行されているエンジンを選択します。
3. 選択したエンジンを実行しているキャッシュが一覧表示されます。
4. キャッシュ名の左側にあるラジオボタンを選択して、変更したいキャッシュを選択します。
5. アクション を選択してから、変更 を選択します。
6. 「使用制限」で、適切なメモリまたはコンピューティング制限を設定します。

7. [プレビュー] をクリックして変更を確認し、[保存] をクリックして変更を保存します。

を使用したスケーリング制限の設定 AWS CLI

CLI を使用してスケーリング制限を変更するには、`modify-serverless-cache` API を使用します。

Linux :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \  
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},  
ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

Windows :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^  
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},  
ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

CLI を使用してスケーリング上限を削除する

CLI を使用してスケーリング制限を削除するには、最小制限パラメータと最大制限パラメータを 0 に設定します。

Linux :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \  
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},  
ECPUPerSecond={Minimum=0,Maximum=0}'
```

Windows :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^  
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},  
ECPUPerSecond={Minimum=0,Maximum=0}'
```

Redis ElastiCache の自己設計クラスターのスケーリング

アプリケーションが処理しなければならないデータの量は、一定ではありません。業務の拡大またはまたは通常の変動が発生すると、需要は増加します。キャッシュを自己管理する場合は、需要

のピークに対して十分なハードウェアを用意する必要がありますが、それにより費用が高くなります。Amazon を使用すると、現在の需要に合わせてスケーリング ElastiCache でき、使用した分だけ支払いが発生します。ElastiCache を使用すると、需要に合わせてキャッシュをスケーリングできます。

以下は、実行するスケーリングアクションに適したトピックの検索に役立ちます。

Redis クラスターのスケーリング

アクション	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
スケールイン	クラスターからのノードの削除	Redis (クラスターモードが有効) でのクラスターのスケーリング
スケールアウト	クラスターへのノードの追加	Redis 用のオンラインリシャードニングおよびシャードの再分散 (クラスターモードが有効)
ノードタイプの変更	<p>より大きいノードタイプへ:</p> <ul style="list-style-type: none"> Redis (クラスターモードが無効) の単一ノードクラスターのスケールアップ レプリカを含む Redis クラスターのスケールアップ <p>より小さいノードタイプへ:</p> <ul style="list-style-type: none"> 単一ノード Redis クラスターのスケールダウン レプリカを含む Redis クラスターのスケールダウン 	ノードタイプの変更によるオンライン垂直スケーリング

アクション	Redis (クラスターモードが無効)	Redis (クラスターモードが有効)
ノードグループの数の変更	Redis (クラスターモードが無効) クラスターではサポートされていません	Redis (クラスターモードが有効) でのクラスターのスケールリング

トピック

- [Redis \(クラスターモードが無効\) クラスターのスケールリング](#)
- [Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)

Redis (クラスターモードが無効) クラスターのスケーリング

Redis (クラスターモードが無効) クラスターは、シャードを持たない単一ノードクラスターでも、1つのシャードを持つマルチノードクラスターでもかまいません。単一ノードクラスターでは、読み取りと書き込みの両方に1つのノードを使用します。マルチノードクラスターには、0~5の読み取り専用レプリカノードのある読み取り/書き込みプライマリノードとして常に1個のノードがあります。

目次

- [Redis \(クラスターモードが無効\) の単一ノードクラスターのスケーリング](#)
 - [Redis \(クラスターモードが無効\) の単一ノードクラスターのスケーリング](#)
 - [Redis \(クラスターモードが無効\) の単一ノードクラスターのスケーリング \(コンソール\)](#)
 - [単一ノード Redis キャッシュクラスターのスケーリング \(AWS CLI\)](#)
 - [単一ノード Redis キャッシュクラスターのスケーリング \(ElastiCache API\)](#)
 - [単一ノード Redis クラスターのスケーリング](#)
 - [単一ノード Redis クラスターのスケーリング \(コンソール\)](#)
 - [単一ノード Redis キャッシュクラスターのスケーリング \(AWS CLI\)](#)
 - [単一ノード Redis キャッシュクラスターのスケーリング \(ElastiCache API\)](#)
- [レプリカノードを含む Redis \(クラスターモードが無効\) クラスターのスケーリング](#)
 - [レプリカを含む Redis クラスターのスケーリング](#)
 - [レプリカを含む Redis クラスターのスケーリング](#)
 - [読み込みキャパシティの増加](#)
 - [読み込みキャパシティの削減](#)

Redis (クラスターモードが無効) の単一ノードクラスターのスケーリング

Redis (クラスターモードが無効) ノードには、すべてのキャッシュのデータと Redis のオーバーヘッドを保存するのに十分なデータ容量が必要です。Redis (クラスターモードが無効) クラスターのデータ容量を変更するには、垂直スケーリングを行う (より大きいノードタイプにスケールアップしてデータ容量を増やすか、より小さいノードタイプにスケールダウンしてデータ容量を減らす) 必要があります。

ElastiCache for Redis のスケールアッププロセスは、既存のデータをベストエフォートで保持するように設計されており、Redis レプリケーションが正常に実行される必要があります。Redis (クラスターモードが無効) クラスターの場合、Redis 用に十分なメモリを確保することをお勧めします。

複数の Redis (クラスターモードが無効) クラスター間でデータを分割することはできません。ただし、クラスターの読み込みキャパシティーを増減させる必要がある場合は、レプリカノードを含む Redis (クラスターモードが無効) クラスターを作成して、リードレプリカを追加または削除できます。プライマリクラスターとして単一ノード Redis キャッシュクラスターを使用して、レプリカノードを含む Redis (クラスターモードが無効) クラスターを作成するには、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」を参照してください。

レプリカを含むクラスターを作成したら、リードレプリカを追加することで、読み込みキャパシティーを増やすことができます。後で必要に応じて、リードレプリカを削除することで、読み込みキャパシティーを減らすことができます。詳細については、「[読み込みキャパシティーの増加](#)」または「[読み込みキャパシティーの削減](#)」を参照してください。

読み込みキャパシティーのスケールアップが可能に加えて、レプリカを含む Redis (クラスターモードが無効) クラスターには、そのほかにもビジネス上の利点があります。詳細については、「[レプリケーショングループを使用する高可用性](#)」を参照してください。

Important

パラメータグループが reserved-memory を使用して Redis のオーバーヘッド用のメモリを確保する場合、スケールアップを開始する前に、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。または、reserved-memory-percent を使用するようにカスタムパラメータグループを変更し、新しいクラスターに対して、パラメータグループを使用することができます。reserved-memory-percent を使用している場合、これを行う必要はありません。詳細については、「[予約メモリの管理](#)」を参照してください。

トピック

- [Redis \(クラスターモードが無効\) の単一ノードクラスターのスケールアップ](#)
- [単一ノード Redis クラスターのスケールダウン](#)

Redis (クラスターモードが無効) の単一ノードクラスターのスケールアップ

単一ノード Redis クラスターをスケールアップするとき、ElastiCache コンソール、AWS CLI、ElastiCache API のいずれを使用する場合でも、によって以下のプロセスが実行されます。

1. 新しいノードタイプの新しいキャッシュクラスターは既存のキャッシュクラスターと同じアベイラビリティゾーンでスピンアップされます。
2. 既存のキャッシュクラスターのキャッシュデータは新しいキャッシュクラスターにコピーされます。このプロセスの所要時間はノードタイプとキャッシュクラスターのデータ量によって異なります。
3. 読み取りと書き込みは、新しいキャッシュクラスターを使用して行われるようになります。新しいキャッシュクラスターのエンドポイントは、古いキャッシュクラスターのもと同じなので、アプリケーションのエンドポイントを更新する必要はありません。DNS エントリが更新されている間、プライマリノードからの読み書きが短時間 (数秒) 中断されるのが分かります。
4. ElastiCache によって古いキャッシュクラスターが削除されます。古いノードへの接続が切断されるため、古いノードからの読み書きが短時間 (数秒) 中断されるのが分かります。

Note

r6gd ノードタイプを実行するクラスターでは、r6gd ノードファミリー内のノードサイズにのみスケールできます。

以下の表に示しているように、次のメンテナンス期間にエンジンのアップグレードがスケジュールされている場合、Redis のスケールアップオペレーションはブロックされます。メンテナンス期間の詳細については、「[メンテナンスの管理](#)」を参照してください。

ブロックされた Redis オペレーション

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

保留中のオペレーションによってブロックされている場合は、以下のいずれかを行うことができます。

- 次のメンテナンス期間に Redis スケールアップオペレーションをスケジュールします。そのためには、[Apply immediately] チェックボックスをオフにします (CLI では `--no-apply-immediately`、API では `ApplyImmediately=false` を使用)。
- Redis のスケールアップオペレーションを実行する次のメンテナンス期間 (またはその後) まで待ちます。
- [Apply Immediately] チェックボックスをオンにして、このキャッシュクラスターの変更に Redis エンジンのアップグレードを追加します (CLI では `--apply-immediately`、API では `ApplyImmediately=true` を使用)。これにより、エンジンのアップグレードがすぐに実行されて、スケールアップオペレーションのブロックが解除されます。

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、単一ノード Redis (クラスターモードが無効) クラスターをスケールアップできます。

Important

パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、スケーリングを開始する前に、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。または、`reserved-memory-percent` を使用するようにカスタムパラメータグループを変更し、新しいクラスターに対して、パラメータグループを使用することができます。`reserved-memory-percent` を使用している場合、これを行う必要はありません。詳細については、「[予約メモリの管理](#)」を参照してください。

Redis (クラスターモードが無効) の単一ノードクラスターのスケールアップ (コンソール)

以下の手順では、ElastiCache マネジメントコンソールを使用して、単一ノード Redis クラスターをスケールアップする方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

単一ノード Redis クラスター をスケールアップするには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。

3. クラスターのリストから、スケールアップするクラスターを選択します (Clustered Redis エンジンではなく Redis エンジンを実行している必要があります)。
4. [Modify] (変更) を選択します。
5. [Modify Cluster] ウィザードで:
 - a. [Node type] リストから、スケールアップするノードタイプを選択します。
 - b. reserved-memory を使用してメモリを管理している場合、[Parameter Group] リストから新しいノードタイプのために適切な容量のメモリを確保するカスタムパラメータグループを選択します。
6. スケールアッププロセスをすぐに実行する場合は、[Apply immediately] ボックスを選択します。[Apply immediately] ボックスを選択していない場合、スケールアッププロセスはこのクラスターの次のメンテナンス期間中に実行されます。
7. [Modify] (変更) を選択します。

前の手順で [Apply immediately] を選択した場合、クラスターのステータスは [modifying] に変わります。ステータスが [available] に変わると、変更は完了し、新しいクラスターの使用を開始できます。

単一ノード Redis キャッシュクラスターのスケールアップ(AWS CLI)

以下の手順では、AWS CLI を使用して単一ノード Redis キャッシュクラスターをスケールアップする方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

単一ノード Redis キャッシュクラスターをスケールアップするには (AWS CLI)

1. 以下のパラメータを指定して AWS CLI `list-allowed-node-type-modifications` コマンドを実行することで、スケールアップできるノードタイプを調べます。

- `--cache-cluster-id`

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --cache-cluster-id my-cache-cluster-id
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^  
  --cache-cluster-id my-cache-cluster-id
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ScaleUpModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",  
    "cache.m3.xlarge",  
    "cache.m4.10xlarge",  
    "cache.m4.2xlarge",  
    "cache.m4.4xlarge",  
    "cache.m4.large",  
    "cache.m4.xlarge",  
    "cache.r3.2xlarge",  
    "cache.r3.4xlarge",  
    "cache.r3.8xlarge",  
    "cache.r3.large",  
    "cache.r3.xlarge"  
  ],  
  "ScaleDownModifications": [  
    "cache.t2.micro",  
    "cache.t2.small",  
    "cache.t2.medium",  
    "cache.t1.small"  
  ],  
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

2. AWS CLI `modify-cache-cluster` コマンドで以下のパラメータを使用して、スケールアップするキャッシュクラスターと新しいより大きいノードタイプを指定することで、既存のキャッシュクラスターを変更します。
 - `--cache-cluster-id` – スケールアップするキャッシュクラスターの名前。

- `--cache-node-type` – キャッシュクラスターのスケールアップ後の新しいノードタイプ。この値は、ステップ 1 で `list-allowed-node-type-modifications` コマンドによって返されるノードタイプのいずれかである必要があります。
- `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
- `--apply-immediately` – スケールアッププロセスがすぐに適用されるようにします。スケールアッププロセスをクラスターの次のメンテナンス期間に延期するには、`--no-apply-immediately` パラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-redis-cache-cluster \  
  --cache-node-type cache.m3.xlarge \  
  --cache-parameter-group-name redis32-m2-xl \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^  
  --cache-cluster-id my-redis-cache-cluster ^  
  --cache-node-type cache.m3.xlarge ^  
  --cache-parameter-group-name redis32-m2-xl ^  
  --apply-immediately
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "CacheCluster": {  
    "Engine": "redis",  
    "CacheParameterGroup": {  
      "CacheNodeIdsToReboot": [],  
      "CacheParameterGroupName": "default.redis6.x",  
      "ParameterApplyStatus": "in-sync"  
    },  
  },  
}
```

```
"SnapshotRetentionLimit": 1,
"CacheClusterId": "my-redis-cache-cluster",
"CacheSecurityGroups": [],
"NumCacheNodes": 1,
"SnapshotWindow": "00:00-01:00",
"CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
"AutoMinorVersionUpgrade": true,
"CacheClusterStatus": "modifying",
"PreferredAvailabilityZone": "us-west-2a",
"ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
"CacheSubnetGroupName": "default",
"EngineVersion": "6.0",
"PendingModifiedValues": {
  "CacheNodeType": "cache.m3.2xlarge"
},
"PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
"CacheNodeType": "cache.m3.medium",
"DataTiering": "disabled"
}
}
```

詳細については、AWS CLI リファレンスの「[modify-cache-cluster](#)」を参照してください。

3. `--apply-immediately` を使用した場合は、以下のパラメータを指定して AWS CLI `describe-cache-clusters` コマンドを使用することで、新しいキャッシュクラスターのステータスを確認します。ステータスが `[available]` に変わると、新しいより大きいキャッシュクラスターの使用を開始できます。
 - `--cache-cluster-id` – 単一ノード Redis のキャッシュクラスターの名前。すべてのキャッシュクラスターではなく特定のキャッシュクラスターの定義を表示するには、このパラメータを使用します。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

詳細については、AWS CLI リファレンスの「[describe-cache-clusters](#)」を参照してください。

単一ノード Redis キャッシュクラスターのスケールアップ (ElastiCache API)

以下の手順では、ElastiCache API を使用して単一ノード Redis キャッシュクラスターをスケールアップする方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

単一ノード Redis キャッシュクラスターをスケールアップするには (ElastiCache API)

1. 以下のパラメータを使用して ElastiCache API `ListAllowedNodeTypeModifications` アクションを実行することで、スケールアップできるノードタイプを調べます。

- `CacheClusterId` – スケールアップする単一ノード Redis キャッシュクラスターの名前。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListAllowedNodeTypeModifications  
&CacheClusterId=MyRedisCacheCluster  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ListAllowedNodeTypeModifications](#)」を参照してください。

2. `ModifyCacheCluster` ElastiCache API アクションと以下のパラメータを使用して、スケールアップするキャッシュクラスターと新しいより大きいノードタイプを指定することで、既存のキャッシュクラスターを変更します。

- `CacheClusterId` – スケールアップするキャッシュクラスターの名前。
- `CacheNodeType` – キャッシュクラスターのスケールアップ後の新しいより大きいノードタイプ。この値は、手順 1 で `ListAllowedNodeTypeModifications` アクションによって返されるノードタイプのいずれかであることが必要です。
- `CacheParameterGroupName` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。

- `ApplyImmediately` – スケールアッププロセスがすぐに実行されるようにするには、`true` に設定します。スケールアッププロセスをクラスターの次のメンテナンス期間に延期するには、`ApplyImmediately=false` パラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyCacheCluster  
&ApplyImmediately=true  
&CacheClusterId=MyRedisCacheCluster  
&CacheNodeType=cache.m3.xlarge  
&CacheParameterGroupName redis32-m2-x1  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ModifyCacheCluster](#)」を参照してください。

3. `ApplyImmediately=true` を使用した場合は、以下のパラメータを指定して ElastiCache API `DescribeCacheClusters` アクションを使用することで、新しいキャッシュクラスターのステータスを確認します。ステータスが `[available]` に変わると、新しいより大きいキャッシュクラスターの使用を開始できます。

- `CacheClusterId` – 単一ノード Redis のキャッシュクラスターの名前。すべてのキャッシュクラスターではなく特定のキャッシュクラスターの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterId=MyRedisCacheCluster  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[DescribeCacheClusters](#)」を参照してください。

単一ノード Redis クラスターのスケールダウン

以下のセクションでは、単一ノード Redis クラスターをより小さいノードタイプにスケールダウンする方法について説明します。新しいより小さいノードタイプがデータと Redis オーバーヘッドのすべてのニーズを満たすのに十分な容量であることを確認するのは、新しいクラスターを長期にわたり適切に運用するために重要です。詳細については、「[Redis スナップショットを作成するのに十分なメモリがあることの確認](#)」を参照してください。

Note

r6gd ノードタイプを実行するクラスターでは、r6gd ノードファミリー内のノードサイズのみスケールできます。

トピック

- [単一ノード Redis クラスターのスケールダウン \(コンソール\)](#)
- [単一ノード Redis キャッシュクラスターのスケールダウン \(AWS CLI\)](#)
- [単一ノード Redis キャッシュクラスターのスケールダウン \(ElastiCache API\)](#)

単一ノード Redis クラスターのスケールダウン (コンソール)

以下の手順では、ElastiCache コンソールを使用して単一ノード Redis クラスターをより小さいノードタイプにスケールダウンする方法について説明しています。

Important

パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、スケーリングを開始する前に、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。または、`reserved-memory-percent` を使用するようにカスタムパラメータグループを変更し、新しいクラスターに対して、パラメータグループを使用することができます。`reserved-memory-percent` を使用している場合、これを行う必要はありません。詳細については、「[予約メモリの管理](#)」を参照してください。

単一ノード Redis クラスターをスケールダウンするには (コンソール)

1. より小さいノードタイプがデータとオーバーヘッドのニーズを満たしていることを確認します。

2. パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。

または、`reserved-memory-percent` を使用するよう、カスタムパラメータグループを変更できます。詳細については、「[予約メモリの管理](#)」を参照してください。

3. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
4. クラスターのリストから、スケールダウンするクラスターを選択します。このクラスターは、Clustered Redis エンジンではなく Redis エンジンを実行している必要があります。
5. [Modify] (変更) を選択します。
6. [Modify Cluster] ウィザードで:
 - a. [ノードのタイプ] リストから、スケールダウンするノードタイプを選択します。
 - b. `reserved-memory` を使用してメモリを管理している場合、[Parameter Group] リストから新しいノードタイプのために適切な容量のメモリを確保するカスタムパラメータグループを選択します。
7. スケールダウンプロセスをすぐに実行する場合は、[すぐに適用] チェックボックスをオンにします。[すぐに適用] チェックボックスを選択しないままにすると、スケールダウンプロセスはこのクラスターの次のメンテナンスウィンドウ中に実行されます。
8. [Modify] (変更) を選択します。
9. クラスターのステータスが [modifying] から [available] に変わると、クラスターは新しいノードタイプにスケールリングされます。アプリケーションでエンドポイントを更新する必要はありません。

単一ノード Redis キャッシュクラスターのスケールダウン (AWS CLI)

次の手順では、AWS CLI を使用して単一ノード Redis キャッシュクラスターをスケールダウンする方法について説明します。

単一ノード Redis キャッシュクラスターをスケールダウンするには (AWS CLI)

1. 以下のパラメータを指定して AWS CLI `list-allowed-node-type-modifications` コマンドを実行することで、スケールダウンできるノードタイプを調べます。
 - `--cache-cluster-id`

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --cache-cluster-id my-cache-cluster-id
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^  
  --cache-cluster-id my-cache-cluster-id
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ScaleUpModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",  
    "cache.m3.xlarge",  
    "cache.m4.10xlarge",  
    "cache.m4.2xlarge",  
    "cache.m4.4xlarge",  
    "cache.m4.large",  
    "cache.m4.xlarge",  
    "cache.r3.2xlarge",  
    "cache.r3.4xlarge",  
    "cache.r3.8xlarge",  
    "cache.r3.large",  
    "cache.r3.xlarge"  
  ],  
  "ScaleDownModifications": [  
    "cache.t2.micro",  
    "cache.t2.small",  
    "cache.t2.medium",  
    "cache.t1.small"  
  ],  
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

2. AWS CLI `modify-cache-cluster` コマンドで以下のパラメータを使用して、スケールダウンするキャッシュクラスターと新しいより小さいノードタイプを指定することで、既存のキャッシュクラスターを変更します。

- `--cache-cluster-id` – スケールダウンするキャッシュクラスターの名前。
- `--cache-node-type` – キャッシュクラスターのスケールアップ後の新しいノードタイプ。この値は、ステップ 1 で `list-allowed-node-type-modifications` コマンドによって返されるノードタイプのいずれかであることが必要です。
- `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
- `--apply-immediately` – スケールダウンプロセスをすぐに適用します。スケールアッププロセスをクラスターの次のメンテナンス期間に延期するには、`--no-apply-immediately` パラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-redis-cache-cluster \  
  --cache-node-type cache.m3.xlarge \  
  --cache-parameter-group-name redis32-m2-x1 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^  
  --cache-cluster-id my-redis-cache-cluster ^  
  --cache-node-type cache.m3.xlarge ^  
  --cache-parameter-group-name redis32-m2-x1 ^  
  --apply-immediately
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "CacheCluster": {  
    "Engine": "redis",
```

```
"CacheParameterGroup": {
  "CacheNodeIdsToReboot": [],
  "CacheParameterGroupName": "default.redis6.x",
  "ParameterApplyStatus": "in-sync"
},
"SnapshotRetentionLimit": 1,
"CacheClusterId": "my-redis-cache-cluster",
"CacheSecurityGroups": [],
"NumCacheNodes": 1,
"SnapshotWindow": "00:00-01:00",
"CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
"AutoMinorVersionUpgrade": true,
"CacheClusterStatus": "modifying",
"PreferredAvailabilityZone": "us-west-2a",
"ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
"CacheSubnetGroupName": "default",
"EngineVersion": "6.0",
"PendingModifiedValues": {
  "CacheNodeType": "cache.m3.2xlarge"
},
"PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
"CacheNodeType": "cache.m3.medium",
"DataTiering": "disabled"
}
}
```

詳細については、AWS CLI リファレンスの「[modify-cache-cluster](#)」を参照してください。

3. `--apply-immediately` を使用した場合は、以下のパラメータを指定して AWS CLI `describe-cache-clusters` コマンドを使用することで、新しいキャッシュクラスターのステータスを確認します。ステータスが `[available]` に変わると、新しいより大きいキャッシュクラスターの使用を開始できます。

 - `--cache-cache cluster-id` – 単一ノード Redis のキャッシュクラスターの名前。すべてのキャッシュクラスターではなく特定のキャッシュクラスターの定義を表示するには、このパラメータを使用します。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

詳細については、AWS CLI リファレンスの「[describe-cache-clusters](#)」を参照してください。

単一ノード Redis キャッシュクラスターのスケールダウン (ElastiCache API)

次の手順では、ElastiCache API を使用して単一ノード Redis キャッシュクラスターをスケールダウンする方法について説明します。

単一ノード Redis キャッシュクラスターをスケールダウンするには (ElastiCache API)

1. 以下のパラメータを使用して ElastiCache API ListAllowedNodeTypeModifications アクションを実行することで、スケールダウンできるノードタイプを確認します。
 - CacheClusterId – スケールダウンする単一ノード Redis キャッシュクラスターの名前。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListAllowedNodeTypeModifications  
&CacheClusterId=MyRedisCacheCluster  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ListAllowedNodeTypeModifications](#)」を参照してください。

2. ModifyCacheCluster ElastiCache API アクションと以下のパラメータを使用して、スケールアップするキャッシュクラスターと新しいより大きいノードタイプを指定することで、既存のキャッシュクラスターを変更します。
 - CacheClusterId – スケールダウンするキャッシュクラスターの名前。
 - CacheNodeType – キャッシュクラスターのスケールダウン後の新しい、より小さいノードタイプ。この値は、手順 1 で ListAllowedNodeTypeModifications アクションによって返されるノードタイプのいずれかであることが必要です。
 - CacheParameterGroupName – (オプション) reserved-memory を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。reserved-memory-percent を使用している場合は、このパラメータを省略できます。
 - ApplyImmediately – スケールダウンプロセスがすぐに実行するには、true に設定します。スケールアッププロセスをクラスターの次のメンテナンス期間に延期するには、ApplyImmediately=false パラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyCacheCluster  
&ApplyImmediately=true  
&CacheClusterId=MyRedisCacheCluster  
&CacheNodeType=cache.m3.xlarge  
&CacheParameterGroupName redis32-m2-xl  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ModifyCacheCluster](#)」を参照してください。

3. ApplyImmediately=true を使用した場合は、以下のパラメータを指定して ElastiCache API DescribeCacheClusters アクションを使用することで、新しいキャッシュクラスタのステータスを確認します。ステータスが available に変わると、新しいより小さいキャッシュクラスタノードの使用を開始できます。
- CacheClusterId – 単一ノード Redis のキャッシュクラスタの名前。すべてのキャッシュクラスタではなく特定のキャッシュクラスタの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterId=MyRedisCacheCluster  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[DescribeCacheClusters](#)」を参照してください。

レプリカノードを含む Redis (クラスターモードが無効) クラスターのスケーリング

レプリカノード (API/CLI ではレプリケーショングループ) を含む Redis クラスターは、自動フェイルオーバーを備えたマルチ AZ が有効なレプリケーションにより、高可用性を実現します。レプリカノードを含むクラスターは、最大 6 つの Redis ノードの論理的な集合であり、1 つのノード (プライマリ) は読み取りと書き込みの両方のリクエストに対応できます。クラスター内の他のすべてのノードは、プライマリの読み取り専用レプリカです。プライマリに書き込まれたデータはクラスターのすべてのリードレプリカに非同期でレプリケートされます。Redis (クラスターモードが無効) は、複数のクラスター間でのデータのパーティション化をサポートしていないため、Redis (クラスターモードが無効) レプリケーショングループの各ノードには、キャッシュデータセット全体が含まれます。Redis (クラスターモードが有効) クラスターは、最大 500 のシャードにまたがるデータのパーティション化をサポートします。

クラスターのデータ容量を変更するには、より大きいノードタイプにスケールアップするか、より小さいノードタイプにスケールダウンする必要があります。

クラスターの読み込みキャパシティーを変更するには、最大 5 のリードレプリカを追加するか、リードレプリカを削除します。

ElastiCache のスケールアッププロセスは、既存のデータをベストエフォートで保持するように設計されており、Redis レプリケーションが正常に実行される必要があります。レプリカを含む Redis クラスターの場合、Redis 用に十分なメモリを確保することをお勧めします。

関連トピック

- [レプリケーショングループを使用する高可用性](#)
- [レプリケーション: Redis \(クラスターモードが無効\) 対 Redis \(クラスターモードが有効\)](#)
- [マルチ AZ による ElastiCache for Redis のダウンタイムの最小化](#)
- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)

トピック

- [レプリカを含む Redis クラスターのスケールアップ](#)
- [レプリカを含む Redis クラスターのスケールダウン](#)
- [読み込みキャパシティーの増加](#)
- [読み込みキャパシティーの削減](#)

レプリカを含む Redis クラスターのスケールアップ

Amazon ElastiCache では、コンソール、CLI、API を使用した Redis (クラスターモードが無効) レプリケーショングループのスケールアップがサポートされています。

スケールアッププロセスが開始されると、ElastiCache によって以下の処理が実行されます。

1. 新しいノードタイプを使用して、レプリケーショングループを起動します。
2. 現行プライマリノードのすべてのデータを新しいプライマリノードにコピーします。
3. 新しいリードレプリカを新しいプライマリノードと同期させます。
4. 新しいノードを参照するように DNS エントリを更新します。このため、アプリケーションのエンドポイントを更新する必要はありません。Redis 5.0.5 以降では、クラスターがオンラインのまま受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケールアップできます。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。
5. 古いノード (CLI/API: レプリケーショングループ) を削除します。古いノードへの接続が切断されるため、古いノードからの読み書きが短時間 (数秒) 中断されるのが分かります。

このプロセスの所要時間はノードタイプとクラスターのデータ量によって異なります。

以下の表に示しているように、クラスターの次のメンテナンス期間にエンジンのアップグレードがスケジュールされている場合、Redis のスケールアップオペレーションはブロックされます。

ブロックされた Redis オペレーション

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

保留中のオペレーションによってブロックされている場合は、以下のいずれかを行うことができます。

- 次のメンテナンス期間に Redis スケールアップオペレーションをスケジュールします。そのためには、[Apply immediately] チェックボックスをオフにします (CLI では `--no-apply-immediately`、API では `ApplyImmediately=false` を使用)。
- Redis のスケールアップオペレーションを実行する次のメンテナンス期間 (またはその後) まで待ちます。
- [Apply Immediately] チェックボックスをオンにして、このキャッシュクラスターの変更に Redis エンジンのアップグレードを追加します (CLI では `--apply-immediately`、API では `ApplyImmediately=true` を使用)。これにより、エンジンのアップグレードがすぐに実行されて、スケールアップオペレーションのブロックが解除されます。

以下のセクションでは、ElastiCache コンソール、AWS CLI、ElastiCache API を使用して、レプリカを含む Redis のクラスターをスケールアップする方法について説明します。

Important

パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、スケーリングを開始する前に、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。または、`reserved-memory-percent` を使用するようにカスタムパラメータグループを変更し、新しいクラスターに対して、パラメータグループを使用することができます。`reserved-memory-percent` を使用している場合、これを行う必要はありません。詳細については、「[予約メモリの管理](#)」を参照してください。

レプリカを含む Redis クラスターのスケールアップ (コンソール)

より大きいノードタイプへのスケールアップにかかる時間はノードタイプと現在のクラスターのデータ量によって異なります。

以下のプロセスでは、ElastiCache コンソールを使用して、レプリカを含むクラスターをその現在のノードタイプから新しいより大きいノードタイプにスケーリングします。このプロセス中に、DNS エントリが更新されている間、プライマリノードからの他のバージョンの読み取りと書き込みが短時間中断される場合があります。5.0.6 バージョン以降で実行されているノードでは 1 秒未満のダウンタイム、古いバージョンでは数秒のダウンタイムが発生する場合があります。

レプリカを含む Redis クラスターをスケールアップするには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターのリストから、スケールアップするクラスターを選択します。このクラスターは、Clustered Redis エンジンではなく Redis エンジンを実行している必要があります。
4. [Modify] (変更) を選択します。
5. [Modify Cluster] ウィザードで:
 - a. [Node type] リストから、スケーリングするノードタイプを選択します。すべてのノードタイプがスケールダウンできるわけではないことに注意してください。
 - b. reserved-memory を使用してメモリを管理している場合、[Parameter Group] リストから新しいノードタイプのために適切な容量のメモリを確保するカスタムパラメータグループを選択します。
6. スケールアッププロセスをすぐに実行する場合は、[Apply immediately] チェックボックスをオンにします。[Apply immediately] チェックボックスをオフのままにすると、スケールアッププロセスは、このクラスターの次のメンテナンス期間中に実行されます。
7. [Modify] (変更) を選択します。
8. クラスターのステータスが [modifying] から [available] に変わると、クラスターは新しいノードタイプにスケーリングされます。アプリケーションでエンドポイントを更新する必要はありません。

Redis レプリケーショングループのスケールアップ (AWS CLI)

次のプロセスでは、AWS CLI を使用して、レプリケーショングループを現在のノードタイプから新しいより大きいノードタイプにスケールします。このプロセスでは、ElastiCache for Redis は DNS エントリを更新し、新しいノードを参照します。このため、アプリケーションのエンドポイントを更新する必要はありません。Redis 5.0.5 以降では、クラスターがオンラインのまま受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケーリングできます。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。

より大きいノードタイプへのスケールアップにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

Redis レプリケーショングループをスケールアップするには (AWS CLI)

1. 以下のパラメータを使用して AWS CLI `list-allowed-node-type-modifications` コマンドを実行することで、スケールアップできるノードタイプを特定します。
 - `--replication-group-id` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --replication-group-id my-repl-group
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^  
  --replication-group-id my-repl-group
```

このオペレーションからの出力は以下のような JSON 形式になります。

```
{  
  "ScaleUpModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",  
    "cache.m3.xlarge",  
    "cache.m4.10xlarge",  
    "cache.m4.2xlarge",  
    "cache.m4.4xlarge",  
    "cache.m4.large",  
    "cache.m4.xlarge",  
    "cache.r3.2xlarge",  
    "cache.r3.4xlarge",  
    "cache.r3.8xlarge",  
    "cache.r3.large",  
    "cache.r3.xlarge"  
  ]  
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

- 以下のパラメータを指定して AWS CLI `modify-replication-group` コマンドを使用することで、現在のレプリケーショングループを新しいノードタイプにスケールアップします。
 - `--replication-group-id` – レプリケーショングループの名前。
 - `--cache-node-type` – このレプリケーショングループのキャッシュクラスターの新しいより大きいノードタイプ。この値は、手順 1 で `list-allowed-node-type-modifications` コマンドによって返されるインスタンスタイプのいずれかであることが必要です。
 - `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
 - `--apply-immediately` – スケールアッププロセスがすぐに適用されるようにします。スケールアップオペレーションを次のメンテナンス期間に延期するには、`--no-apply-immediately` を使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id my-repl-group \  
  --cache-node-type cache.m3.xlarge \  
  --cache-parameter-group-name redis32-m3-2x1 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id my-repl-group ^  
  --cache-node-type cache.m3.xlarge ^  
  --cache-parameter-group-name redis32-m3-2x1 \  
  --apply-immediately
```

このコマンドからの出力は以下のような JSON 形式になります。

```
{
  "ReplicationGroup": {
    "Status": "available",
    "Description": "Some description",
    "NodeGroups": [{
      "Status": "available",
      "NodeGroupMembers": [{
        "CurrentRole": "primary",
        "PreferredAvailabilityZone": "us-west-2b",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Port": 6379,
          "Address": "my-repl-group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
        }
      }],
      "CacheClusterId": "my-repl-group-001"
    },
    {
      "CurrentRole": "replica",
      "PreferredAvailabilityZone": "us-west-2c",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
        "Port": 6379,
        "Address": "my-repl-group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
      }
    },
    {
      "CacheClusterId": "my-repl-group-002"
    }
  ],
  "NodeGroupId": "0001",
  "PrimaryEndpoint": {
    "Port": 6379,
    "Address": "my-repl-group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
  }
},
"ReplicationGroupId": "my-repl-group",
"SnapshotRetentionLimit": 1,
"AutomaticFailover": "disabled",
"SnapshotWindow": "12:00-13:00",
"SnapshottingClusterId": "my-repl-group-002",
"MemberClusters": [
  "my-repl-group-001",
  "my-repl-group-002"
],
}
```

```
"PendingModifiedValues": {}  
}  
}
```

詳細については、AWS CLI リファレンスの「[modify-replication-group](#)」を参照してください。

3. `--apply-immediately` パラメータを使用した場合、以下のパラメータを使用して AWS CLI `describe-replication-group` コマンドを使用することで、レプリケーショングループのステータスをモニタリングします。ステータスはまだ [変更中] ですが、5.0.6 バージョン以降で実行されているノードのダウンタイムは 1 秒未満であり、DNS エントリが更新されている間、プライマリノードからの古いバージョンの読み取りと書き込みが短時間中断する場合があります。

- `--replication-group-id` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-replication-groups \  
  --replication-group-id my-replication-group
```

Windows の場合:

```
aws elasticache describe-replication-groups ^  
  --replication-group-id my-replication-group
```

詳細については、AWS CLI リファレンスの「[describe-replication-groups](#)」を参照してください。

Redis レプリケーショングループのスケールアップ (ElastiCache API)

以下のプロセスでは、ElastiCache API を使用して、レプリケーショングループをその現在のノードタイプから新しいより大きいノードタイプにスケールアップします。Redis 5.0.5 以降では、クラスターがオンラインのままで受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケールアップできます。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。

より大きいノードタイプへのスケールアップにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

Redis レプリケーショングループをスケールアップするには (ElastiCache API)

1. 以下のパラメータを指定して ElastiCache API `ListAllowedNodeTypeModifications` アクションを使用することで、スケールアップできるノードタイプを調べます。
 - `ReplicationGroupId` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListAllowedNodeTypeModifications  
&ReplicationGroupId=MyReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ListAllowedNodeTypeModifications](#)」を参照してください。

2. 以下のパラメータを指定して `ModifyRedplicationGroup` ElastiCache API アクションを使用することで、現在のレプリケーショングループを新しいノードタイプにスケールアップします。
 - `ReplicationGroupId` – レプリケーショングループの名前。
 - `CacheNodeType` – このレプリケーショングループのキャッシュクラスターの新しいより大きいノードタイプ。この値は、手順 1 で `ListAllowedNodeTypeModifications` アクションによって返されるインスタンスタイプのいずれかであることが必要です。
 - `CacheParameterGroupName` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
 - `ApplyImmediately` – スケールアッププロセスがすぐに適用されるようにするには、`true` に設定します。スケールアッププロセスを次のメンテナンス期間に延期するには、`ApplyImmediately=false` を使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyReplicationGroup
```



```
&ApplyImmediately=true
&CacheNodeType=cache.m3.2xlarge
&CacheParameterGroupName=redis32-m3-2x1
&ReplicationGroupId=myReplGroup
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&Version=2014-12-01
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

詳細については、Amazon ElastiCache API リファレンスの「[ModifyReplicationGroup](#)」を参照してください。

3. ApplyImmediately=true を使用した場合、以下のパラメータを指定して ElastiCache API DescribeReplicationGroups アクションを使用することで、レプリケーショングループのステータスをモニタリングします。ステータスが [modifying] から [available] に変わると、スケールアップした新しいレプリケーショングループへの書き込みを開始できます。
- ReplicationGroupId – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReplicationGroups
&ReplicationGroupId=MyReplGroup
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[DescribeReplicationGroups](#)」を参照してください。

レプリカを含む Redis クラスターのスケールダウン

以下のセクションでは、レプリカノードを含む Redis (クラスターモードが無効) キャッシュクラスターをより小さいノードタイプにスケールダウンする方法について説明します。新しいより小さいノードタイプがデータとオーバーヘッドのすべてのニーズを満たすのに十分な容量であることを確認するのは、新しいクラスターを長期にわたり適切に運用するために重要です。詳細については、「[Redis スナップショットを作成するのに十分なメモリがあることの確認](#)」を参照してください。

Note

r6gd ノードタイプを実行するクラスターでは、r6gd ノードファミリー内のノードサイズにのみスケールできます。

Important

パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、スケーリングを開始する前に、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。または、`reserved-memory-percent` を使用するようにカスタムパラメータグループを変更し、新しいクラスターに対して、パラメータグループを使用することができます。`reserved-memory-percent` を使用している場合、これを行う必要はありません。詳細については、「[予約メモリの管理](#)」を参照してください。

レプリカを含む Redis クラスターのスケールダウン (コンソール)

以下のプロセスでは、ElastiCache コンソールを使用して、レプリカノードを含む Redis クラスターをより小さいノードタイプにスケーリングします。

レプリカノードを含む Redis のクラスターをスケールダウンするには (コンソール)

- より小さいノードタイプがデータとオーバーヘッドのニーズを満たしていることを確認します。
- パラメータグループが `reserved-memory` を使用して Redis のオーバーヘッド用のメモリを確保する場合、新しいノードタイプ用に適切な容量のメモリを確保するカスタムパラメータグループがあることを確認してください。

または、`reserved-memory-percent` を使用するよう、カスタムパラメータグループを変更できます。詳細については、「[予約メモリの管理](#)」を参照してください。

3. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
4. クラスターのリストから、スケールダウンするクラスターを選択します。このクラスターは、Clustered Redis エンジンではなく Redis エンジンを実行している必要があります。
5. [Modify] (変更) を選択します。
6. [Modify Cluster] ウィザードで:
 - a. [ノードのタイプ] リストから、スケールダウンするノードタイプを選択します。
 - b. reserved-memory を使用してメモリを管理している場合、[Parameter Group] リストから新しいノードタイプのために適切な容量のメモリを確保するカスタムパラメータグループを選択します。
7. スケールダウンプロセスをすぐに実行する場合は、[すぐに適用] チェックボックスをオンにします。[すぐに適用] チェックボックスを選択しないままにすると、スケールダウンプロセスはこのクラスターの次のメンテナンスウィンドウ中に実行されます。
8. [Modify] (変更) を選択します。
9. クラスターのステータスが [modifying] から [available] に変わると、クラスターは新しいノードタイプにスケールリングされます。アプリケーションでエンドポイントを更新する必要はありません。

Redis レプリケーショングループのスケールダウン (AWS CLI)

次のプロセスでは、AWS CLI を使用して、レプリケーショングループを現在のノードタイプから新しいより小さいノードタイプにスケールします。このプロセスでは、ElastiCache for Redis は DNS エントリを更新し、新しいノードを参照します。このため、アプリケーションのエンドポイントを更新する必要はありません。Redis 5.0.5 以降では、クラスターがオンラインのまま受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケールリングできます。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。

ただし、リードレプリカキャッシュクラスターからの読み取りは中断されません。

より小さいノードタイプへのスケールダウンにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

Redis レプリケーショングループをスケールダウンするには (AWS CLI)

1. 次のパラメータを使用して AWS CLI `list-allowed-node-type-modifications` コマンドを実行することで、スケールダウンできるノードタイプを確認します。
 - `--replication-group-id` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --replication-group-id my-repl-group
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^  
  --replication-group-id my-repl-group
```

このオペレーションからの出力は以下のような JSON 形式になります。

```
{  
  "ScaleDownModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",  
    "cache.m3.xlarge",  
    "cache.m4.10xlarge",  
    "cache.m4.2xlarge",  
    "cache.m4.4xlarge",  
    "cache.m4.large",  
    "cache.m4.xlarge",  
    "cache.r3.2xlarge",  
    "cache.r3.4xlarge",  
    "cache.r3.8xlarge",  
    "cache.r3.large",  
    "cache.r3.xlarge"  
  ]  
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

- 以下のパラメータを指定して AWS CLI `modify-replication-group` コマンドを使用することで、現在のレプリケーショングループを新しいノードタイプにスケールアップします。
 - `--replication-group-id` – レプリケーショングループの名前。
 - `--cache-node-type` – このレプリケーショングループのキャッシュクラスターの新しいより小さいノードタイプ。この値は、手順 1 で `list-allowed-node-type-modifications` コマンドによって返されるインスタンスタイプのいずれかであることが必要です。
 - `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
 - `--apply-immediately` – スケールアッププロセスがすぐに適用されるようにします。スケールアップオペレーションを次のメンテナンス期間に延期するには、`--no-apply-immediately` を使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id my-repl-group \  
  --cache-node-type cache.t2.small \  
  --cache-parameter-group-name redis32-m3-2x1 \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id my-repl-group ^  
  --cache-node-type cache.t2.small ^  
  --cache-parameter-group-name redis32-m3-2x1 \  
  --apply-immediately
```

このコマンドからの出力は以下のような JSON 形式になります。

```
{"ReplicationGroup": {
  "Status": "available",
  "Description": "Some description",
  "NodeGroups": [
    {
      "Status": "available",
      "NodeGroupMembers": [
        {
          "CurrentRole": "primary",
          "PreferredAvailabilityZone": "us-west-2b",
          "CacheNodeId": "0001",
          "ReadEndpoint": {
            "Port": 6379,
            "Address": "my-repl-
group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
          },
          "CacheClusterId": "my-repl-group-001"
        },
        {
          "CurrentRole": "replica",
          "PreferredAvailabilityZone": "us-west-2c",
          "CacheNodeId": "0001",
          "ReadEndpoint": {
            "Port": 6379,
            "Address": "my-repl-
group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
          },
          "CacheClusterId": "my-repl-group-002"
        }
      ],
      "NodeGroupId": "0001",
      "PrimaryEndpoint": {
        "Port": 6379,
        "Address": "my-repl-
group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
      }
    }
  ],
  "ReplicationGroupId": "my-repl-group",
  "SnapshotRetentionLimit": 1,
  "AutomaticFailover": "disabled",
  "SnapshotWindow": "12:00-13:00",
```

```
    "SnapshottingClusterId": "my-repl-group-002",
    "MemberClusters": [
      "my-repl-group-001",
      "my-repl-group-002",
    ],
    "PendingModifiedValues": {}
  }
}
```

詳細については、AWS CLI リファレンスの「[modify-replication-group](#)」を参照してください。

3. `--apply-immediately` パラメータを使用した場合、以下のパラメータを使用して AWS CLI `describe-replication-group` コマンドを使用することで、レプリケーショングループのステータスをモニタリングします。ステータスが `modifying` から `available` に変わると、スケールダウンした新しいレプリケーショングループへの書き込みを開始できます。
 - `--replication-group-id` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-replication-group \  
  --replication-group-id my-replication-group
```

Windows の場合:

```
aws elasticache describe-replication-groups ^  
  --replication-group-id my-replication-group
```

詳細については、AWS CLI リファレンスの「[describe-replication-groups](#)」を参照してください。

Redis レプリケーショングループのスケールダウン (ElastiCache API)

以下のプロセスでは、ElastiCache API を使用して、レプリケーショングループをその現在のノードタイプから新しいより小さいノードタイプにスケールリングします。このプロセスでは、ElastiCache for Redis は DNS エントリを更新し、新しいノードを参照します。このため、アプリケーションのエンドポイントを更新する必要はありません。Redis 5.0.5 以降では、クラスターがオンラインのまま

で受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケーリングできません。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。ただし、リードレプリカキャッシュクラスターからの読み取りは中断されません。

より小さいノードタイプへのスケールダウンにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

Redis レプリケーショングループをスケールダウンするには (ElastiCache API)

1. 以下のパラメータを指定して ElastiCache API `ListAllowedNodeTypeModifications` アクションを使用することで、スケールダウンできるノードタイプを調べます。
 - `ReplicationGroupId` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListAllowedNodeTypeModifications  
&ReplicationGroupId=MyReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[ListAllowedNodeTypeModifications](#)」を参照してください。

2. 以下のパラメータを指定して `ModifyRedplicationGroup` ElastiCache API アクションを使用することで、現在のレプリケーショングループを新しいノードタイプにスケールアップします。
 - `ReplicationGroupId` – レプリケーショングループの名前。
 - `CacheNodeType` – このレプリケーショングループのキャッシュクラスターの新しいより小さいノードタイプ。この値は、手順 1 で `ListAllowedNodeTypeModifications` アクションによって返されるインスタンスタイプのいずれかであることが必要です。
 - `CacheParameterGroupName` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な

容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。reserved-memory-percent を使用している場合は、このパラメータを省略できます。

- ApplyImmediately – スケールアッププロセスがすぐに適用されるようにするには、true に設定します。スケールダウンプロセスを次のメンテナンスウィンドウに延期するには、ApplyImmediately=false を使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=ModifyReplicationGroup  
  &ApplyImmediately=true  
  &CacheNodeType=cache.m3.2xlarge  
  &CacheParameterGroupName=redis32-m3-2x1  
  &ReplicationGroupId=myReplGroup  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &Version=2014-12-01  
  &X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
  &X-Amz-Date=20141201T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

詳細については、Amazon ElastiCache API リファレンスの「[ModifyReplicationGroup](#)」を参照してください。

3. ApplyImmediately=true を使用した場合、以下のパラメータを指定して ElastiCache API DescribeReplicationGroups アクションを使用することで、レプリケーショングループのステータスをモニタリングします。ステータスが modifying から available に変わると、スケールダウンした新しいレプリケーショングループへの書き込みを開始できます。
- ReplicationGroupId – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=DescribeReplicationGroups  
  &ReplicationGroupId=MyReplGroup  
  &Version=2015-02-02
```

```
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[DescribeReplicationGroups](#)」を参照してください。

読み込みキャパシティーの増加

読み込みキャパシティーを増やすには、Redis レプリケーショングループにリードレプリカ (最大 5 個) を追加します。

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、Redis クラスターの読み込みキャパシティーをスケーリングできます。詳細については、「[Redis \(クラスターモードが無効\) レプリケーショングループのリードレプリカの追加](#)」を参照してください。

読み込みキャパシティーの削減

読み込みキャパシティーを減らすには、レプリカを含む Redis クラスター (API/CLI ではレプリケーショングループ) から 1 つ以上のリードレプリカを削除します。クラスターで自動フェイルオーバーを備えたマルチ AZ が有効な場合は、最初にマルチ AZ を無効にしないと、最後のリードレプリカを削除することはできません。詳細については、「[レプリケーショングループの変更](#)」を参照してください。

詳細については、「[Redis \(クラスターモードが無効 \) レプリケーショングループのリードレプリカの削除](#)」を参照してください。

Redis (クラスターモードが有効) でのクラスターのスケールリング

クラスターの需要の変化に応じて Redis (クラスターモードが有効) クラスター内のシャード数を変更することで、パフォーマンスを向上させたりコストを削減したりできます。そのために、スケールリングプロセス中でもクラスターがリクエストを処理し続けることができる、オンライン水平スケールリングの使用をお勧めします。

クラスターを再スケールリングするかどうかの判断条件には、次のようなものがあります。

- **メモリプレッシャー:**

クラスター内のノードがメモリプレッシャーを受けている場合、より多くのリソースがより効率よくデータを保存してリクエストを処理するようにスケールアウトできます。

、 SwapUsageおよび のメトリクスをモニタリングすることでFreeableMemory、 ノードにメモリ負荷がかかっているかどうかを判断できますBytesUseForCache。

- **CPU やネットワークボトルネック:**

レイテンシーやスループットがクラスターの問題となっている場合、問題解決のためにスケールアウトが必要な場合があります。

CPUUtilizationNetworkBytes、 CurrConnections、 および のメトリクスをモニタリングすることで、レイテンシーとスループットレベルをモニタリングできますNewConnections。 NetworkBytes

- **クラスターのサイズが大きすぎます:**

現在のクラスターの需要からすると、スケールインを行ってもパフォーマンスに影響せず、コストも削減できます。

クラスターの使用状況をモニタリングして、 FreeableMemory、 SwapUsage、 CPUUtilization、 NetworkBytesIn BytesUseForCache、 NetworkBytesOut、 CurrConnectionsおよび のメトリクスを使用して安全にスケールインできるかどうかを判断できますNewConnections。

パフォーマンスに対するスケールリングの影響

オフライン処理を使用してスケールリングすると、処理の大部分でクラスターがオフラインになるため、リクエストに対応できなくなります。オンラインメソッドを使用してスケールリングすると、スケールリングは大量の演算を行うオペレーションであるため、パフォーマンスがある程度低下します。その場合でも、クラスターはスケールリングオペレーション全体を通してリクエストに対応しつづけます。エクスペリエンスがどれほど低下するかは、通常の CPU 使用率とデータによって異なります。

Redis (クラスターモードが有効) クラスターをスケーリングするには、2つの方法として水平スケーリングと垂直スケーリングがあります。

- 水平スケーリングでは、ノードグループ (シャード) を追加または削除することで、レプリケーショングループ内のノードグループ (シャード) の数を変更できます。オンラインのリシャードディングプロセスでは、クラスターが着信リクエストの処理を継続しながら、スケールイン/スケールアウトが可能です。

新しいクラスターで、古いクラスターと異なるスロットを設定します。オフラインメソッドに限られます。

- 垂直スケーリング - ノードタイプを変更することで、クラスターのサイズを変更します。オンラインの垂直スケーリングでは、クラスターが着信リクエストの処理を継続しながら、スケールアップ/ダウンが可能です。

スケールインまたはスケールダウンによってクラスターのサイズとメモリ容量を減らす場合は、新しい構成にデータと Redis のオーバーヘッド用の十分なメモリがあることを確認します。

詳細については、「[キャッシュノードサイズの選択](#)」を参照してください。

目次

- [Redis \(クラスターモードが有効\) のオフラインの再分散とシャードの再分散](#)
- [Redis 用のオンラインリシャードディングおよびシャードの再分散 \(クラスターモードが有効\)](#)
 - [オンラインリシャードディングによるシャードの追加](#)
 - [オンラインリシャードディングによるシャードの削除](#)
 - [シャードの削除 \(コンソール\)](#)
 - [シャードの削除 \(AWS CLI\)](#)
 - [シャードの削除 \(ElastiCache API\)](#)
 - [オンラインのシャード再分散](#)
 - [オンラインのシャード再分散 \(コンソール\)](#)
 - [オンラインのシャード再分散 \(AWS CLI\)](#)
 - [オンラインシャードの再調整 \(ElastiCache API\)](#)
- [ノードタイプの変更によるオンライン垂直スケーリング](#)
 - [オンラインスケールアップ](#)
 - [Redis キャッシュクラスターのスケールアップ \(コンソール\)](#)

- [Redis キャッシュクラスターのスケールアップ \(AWS CLI\)](#)
- [Redis キャッシュクラスターのスケールアップ \(ElastiCache API\)](#)
- [オンラインスケールダウン](#)
 - [Redis キャッシュクラスターのスケールダウン \(コンソール\)](#)
 - [単一ノード Redis キャッシュクラスターのスケールダウン \(AWS CLI\)](#)
 - [Redis キャッシュクラスターのスケールダウン \(ElastiCache API\)](#)

Redis (クラスターモードが有効) のオフラインの再分散とシャードの再分散

オフラインのシャード再構成の主な利点は、レプリケーショングループからシャードを追加または削除する以上のことが行えることです。オフラインでリシャーディングすると、レプリケーショングループのシャード数の変更に加えて、次のことを実行できます。

Note

オフラインリシャーディングは、データ階層化が有効になっている Redis クラスターではサポートされません。詳細については、「[データ階層化](#)」を参照してください。

- レプリケーショングループのノードタイプを変更します。
- レプリケーショングループ内の各ノードに、アベイラビリティゾーンを指定します。
- 新しいエンジンバージョンに更新します。
- 各シャード内のレプリカノードの数を個別に指定します。
- 各シャードにキースペースを指定します。

オフラインのシャード再構成の主な欠点は、クラスターが復元処理の開始からオフラインになり、アプリケーションのエンドポイントを更新するまで継続することです。クラスターがオフラインになる時間の長さは、クラスターのデータ量によって変わります。

シャード Redis (クラスターモードが有効) クラスターをオフラインに再設定するには

1. 既存 Redis クラスターの手動バックアップを作成します。詳細については、「[手動バックアップの取得](#)」を参照してください。
2. バックアップから復元して新しいクラスターを作成します。詳細については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。

3. アプリケーション内のエンドポイントを、新しいクラスターのエンドポイントに更新します。詳細については、「[接続エンドポイントの検索](#)」を参照してください。

Redis 用のオンラインリシャーディングおよびシャードの再分散 (クラスターモードが有効)

Amazon ElastiCache for Redis バージョン 3.2.10 以降でオンラインリシャーディングとシャードの再調整を使用すると、ダウンタイムなしで ElastiCache for Redis (クラスターモードが有効) を動的にスケーリングできます。このアプローチでは、クラスターはスケーリングや再分散が処理中でもリクエストに対応し続けることができます。

以下の操作を行うことができます。

- [スケールアウト] – シャード (ノードグループ) を Redis (クラスターモードが有効) クラスター (レプリケーショングループ) に追加することで、読み込みおよび書き込みキャパシティーを増やすことができます。

レプリケーショングループに 1 つ以上のシャードを追加する場合、それぞれの新しいノードのノード数は既存の最小のシャードのシャード数と同じです。

- [スケールイン] – 読み込みおよび書き込みキャパシティーを減らして、Redis (クラスターモードが有効) クラスターからシャードを削除することでコストを削減します。
- 再調整 — ElastiCache for Redis (クラスターモードが有効) クラスター内のシャード間でキースペースを移動し、可能な限りシャード間で均等に分散させます。

次のことはできません。

- シャードを個別に構成:

シャードのキースペースを個別に指定することはできません。これを行うには、オフライン処理を使用する必要があります。

現在、ElastiCache for Redis のオンラインリシャーディングとリバランシングには次の制限が適用されます。

- このプロセスには Redis エンジンバージョン 3.2.10 以降が必要です。エンジンバージョンのアップグレードについての詳細は、「[エンジンバージョンとアップグレード](#)」を参照してください。
- スロットまたはキースペース、および大きなアイテムには制限があります。

シャード内のキーのいずれかに大きなアイテムが含まれる場合、そのキーはスケールアウトまたは再分散の際に移行されません。この機能により、アンバランスなシャードになる可能性があります。

シャード内のキーのいずれかに大きなアイテム (シリアル化後 256 MB より大きいアイテム) が含まれる場合、シャードはスケールイン時に削除されません。この機能により、一部のシャードは削除されない可能性があります。

- スケールアウトの際、新しいシャードのノード数はいずれも、既存の最小のシャードのノード数と等しくなります。
- スケールアウトの際、既存のすべてのシャードに共通するタグは、すべて新しいシャードにコピーされます。
- Global Data Store クラスターをスケールアウトする場合、ElastiCache は既存のノードの 1 つから新しいノードに関数を自動的にレプリケートしません (複数可)。すべてのシャードが同じ関数を持つように、クラスターをスケールアウトした後に、新しいシャードに関数を読み込むことをお勧めします。

Note

ElastiCache for Redis バージョン 7 以降: クラスターをスケールアウトすると、ElastiCache は既存のノードの 1 つにロードされた関数 (ランダムに選択) を新しいノードに自動的にレプリケートします。アプリケーションが [Redis Functions](#) を使用している場合は、スケールアウトする前にすべての関数をすべてのシャードにロードして、for Redis クラスターが異なる ElastiCache シャードで異なる関数定義に陥らないようにすることをお勧めします。

詳細については、「[オンラインクラスターのサイズ変更](#)」を参照してください。

、、 AWS CLI および ElastiCache API を使用して AWS Management Console、ElastiCache for Redis (クラスターモードが有効) クラスターを水平方向にスケーリングまたは再調整できます。

オンラインリシャードイングによるシャードの追加

、、 または ElastiCache API を使用して AWS Management Console AWS CLI、Redis (クラスターモードが有効) クラスターにシャードを追加できます。Redis (クラスターモードが有効) クラスターにシャードを追加すると、既存のシャードのすべてのタグが新しいシャードにコピーされます。

シャードの追加 (コンソール)

を使用して AWS Management Console、1 つ以上のシャードを Redis (クラスターモードが有効) クラスターに追加できます。以下の手順では、このプロセスについて説明します。

シャードを Redis (クラスターモードが有効) に追加するには

1. <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. シャードを追加する Redis (クラスターモードが有効) クラスターの名前 (クラスター名の左にあるボックスではなく) を見つけて選択します。

Tip

Redis (クラスターモードが有効) では、[Mode (モード)] 列に [クラスター化されたRedis] が表示されます。

4. [Add shard] を選択します。
 - a. [追加するシャード数] で、このクラスターに追加するシャード数を選択します。
 - b. [アベイラビリティゾーン] で、[No preference] または [Specify availability zones] を選択します。
 - c. [Specify availability zones] を選択した場合は、各シャードのそれぞれのノードごとに、アベイラビリティゾーンのリストからノードのアベイラビリティゾーンを選択します。
 - d. [追加] を選択します。

シャードの追加 (AWS CLI)

以下のプロセスでは、AWS CLIを使用してシャードを追加し、Redis (クラスターモードが有効) クラスターでシャードの再構成を行う方法について説明します。

`modify-replication-group-shard-configuration` を使って以下のパラメータを使用します。

パラメータ

- `--apply-immediately` – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。

- `--replication-group-id` – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。
- `--node-group-count` – 必須。オペレーションの完了時に存在するシャード (ノードグループ) 数を指定します。シャードを追加する場合、`--node-group-count` の値は現在のシャード数より大きくなければなりません。

オプションで、`--resharding-configuration` を使用してレプリケーショングループ内の各ノードにアベイラビリティゾーンを指定できます。

- `--resharding-configuration` - オプション。レプリケーショングループの各シャードのそれぞれのノードに推奨される、アベイラビリティゾーンのリスト。`--node-group-count` の値が現在のシャード数より大きい場合にのみ、このパラメータを使用します。シャードを追加するときにこのパラメータを省略すると、Amazon ElastiCache は新しいノードのアベイラビリティゾーンを選択します。

次の例では、Redis (クラスターモードが有効) クラスター `my-cluster` の 4 つのシャードでキースペースを再構成します。また、この例では、各シャードでそれぞれのノードのアベイラビリティゾーンを指定しています。オペレーションはすぐに始まります。

Example - シャードの追加

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group-shard-configuration \  
  --replication-group-id my-cluster \  
  --node-group-count 4 \  
  --resharding-configuration \  
    "PreferredAvailabilityZones=us-east-2a,us-east-2c" \  
    "PreferredAvailabilityZones=us-east-2b,us-east-2a" \  
    "PreferredAvailabilityZones=us-east-2c,us-east-2d" \  
    "PreferredAvailabilityZones=us-east-2d,us-east-2c" \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group-shard-configuration ^  
  --replication-group-id my-cluster ^  
  --node-group-count 4 ^  
  --resharding-configuration ^  
    "PreferredAvailabilityZones=us-east-2a,us-east-2c" ^
```

```
"PreferredAvailabilityZones=us-east-2b,us-east-2a" ^  
"PreferredAvailabilityZones=us-east-2c,us-east-2d" ^  
"PreferredAvailabilityZones=us-east-2d,us-east-2c" ^  
--apply-immediately
```

詳細については、AWS CLI ドキュメントの[modify-replication-group-shard-configuration](#)」を参照してください。

シャードの追加 (ElastiCache API)

ElastiCache API を使用して、ModifyReplicationGroupShardConfiguration オペレーションを使用して Redis (クラスターモードが有効) クラスターのシャードをオンラインで再設定できます。

ModifyReplicationGroupShardConfiguration を使って以下のパラメータを使用します。

パラメータ

- ApplyImmediately=true – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。
- ReplicationGroupId – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。
- NodeGroupCount – 必須。オペレーションの完了時に存在するシャード (ノードグループ) 数を指定します。シャードを追加する場合、NodeGroupCount の値は現在のシャード数より大きくなければなりません。

オプションで、ReshardingConfiguration を使用してレプリケーショングループ内の各ノードにアベイラビリティゾーンを指定できます。

- ReshardingConfiguration - オプション。レプリケーショングループの各シャードのそれぞれのノードに推奨される、アベイラビリティゾーンのリスト。NodeGroupCount の値が現在のシャード数より大きい場合にのみ、このパラメータを使用します。シャードを追加するときこのパラメータを省略すると、Amazon は新しいノードのアベイラビリティゾーン ElastiCache を選択します。

以下のプロセスでは、ElastiCache API を使用してシャードを追加することで、Redis (クラスターモードが有効) クラスター内のシャードを再設定する方法について説明します。

Example - シャードの追加

次の例では、Redis (クラスターモードが有効) クラスター `my-cluster` にノードグループを追加します。オペレーション完了時に合計 4 つのノードグループが存在することになります。また、この例では、各シャードでそれぞれのノードのアベイラビリティゾーンを指定しています。オペレーションはすぐに始まります。

```
https://elasticache.us-east-2.amazonaws.com/
  ?Action=ModifyReplicationGroupShardConfiguration
  &ApplyImmediately=true
  &NodeGroupCount=4
  &ReplicationGroupId=my-cluster

  &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2a

  &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2c

  &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2b

  &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2a

  &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2c

  &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2d

  &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2d

  &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2c

  &Version=2015-02-02
  &SignatureVersion=4
  &SignatureMethod=HmacSHA256
  &Timestamp=20171002T192317Z
  &X-Amz-Credential=<credential>
```

詳細については、API リファレンスの[ModifyReplicationGroupShard](#)「[設定 ElastiCache](#)」を参照してください。

オンラインリシャードイングによるシャードの削除

、または ElastiCache API を使用して AWS Management Console AWS CLI、Redis (クラスターモードが有効) クラスターからシャードを削除できます。

トピック

- [シャードの削除 \(コンソール\)](#)
- [シャードの削除 \(AWS CLI\)](#)
- [シャードの削除 \(ElastiCache API\)](#)

シャードの削除 (コンソール)

以下のプロセスでは、AWS Management Consoleを使用してシャードを削除し、Redis (クラスターモードが有効) クラスターでシャードの再構成を行う方法について説明します。

レプリケーショングループからノードグループ (シャード) を削除する前に、はすべてのデータが残りのシャードに収まる ElastiCache ことを確認します。データが収まる場合、指定したシャードはリクエストに応じてレプリケーショングループから削除されます。データが残りのノードグループに収まらない場合、プロセスは終了し、レプリケーショングループはリクエスト前と同じノードグループ設定のままになります。

を使用して AWS Management Console、Redis (クラスターモードが有効) クラスターから 1 つ以上のシャードを削除できます。レプリケーショングループのすべてのシャードを削除することはできません。代わりに、レプリケーショングループを削除する必要があります。詳細については、「[レプリケーショングループの削除](#)」を参照してください。次の手順では、1 つ以上のシャードを削除する手順を説明します。

Redis (クラスターモードが有効) クラスターからシャードを削除するには

1. <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. シャードを削除する Redis (クラスターモードが有効) クラスターの名前 (クラスター名の左にあるボックスではなく) を見つけて選択します。

i Tip

Redis (クラスターモードが有効) クラスターは、[シャード] 列で 1 より大きい値を持ちます。

4. シャードの一覧から、削除する各シャードの名前の左にあるチェックボックスを選択します。
5. [Delete shard] を選択します。

シャードの削除 (AWS CLI)

以下のプロセスでは、AWS CLIを使用してシャードを削除し、Redis (クラスターモードが有効) クラスターでシャードの再構成を行う方法について説明します。

A Important

レプリケーショングループからノードグループ (シャード) を削除する前に、はすべてのデータが残りのシャードに収まる ElastiCache ことを確認します。データが収まる場合、指定されたシャード (--node-groups-to-remove) はリクエストに応じてレプリケーショングループから削除され、キースペースは残りのシャードにマッピングされます。データが残りのノードグループに収まらない場合、プロセスは終了し、レプリケーショングループはリクエスト前と同じノードグループ設定のままになります。

を使用して AWS CLI、Redis (クラスターモードが有効) クラスターから 1 つ以上のシャードを削除できます。レプリケーショングループのすべてのシャードを削除することはできません。代わりに、レプリケーショングループを削除する必要があります。詳細については、「[レプリケーショングループの削除](#)」を参照してください。

modify-replication-group-shard-configuration を使って以下のパラメータを使用します。

パラメータ

- --apply-immediately – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。
- --replication-group-id – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。

- `--node-group-count` – 必須。オペレーションの完了時に存在するシャード (ノードグループ) 数を指定します。シャードを削除する場合、`--node-group-count` の値は現在のシャード数より小さくなければなりません。
- `--node-groups-to-remove` – `--node-group-count` が現在のノードグループ (シャード) 数より少ない場合は必須です。レプリケーショングループから削除するシャード (ノードグループ) ID の一覧。

次の手順では、1 つ以上のシャードを削除する手順を説明します。

Example - シャードの削除

次の例では、Redis (クラスターモードが有効) クラスター `my-cluster` から 2 つのノードグループを削除します。オペレーション完了時に合計 2 つのノードグループが存在することになります。削除されたシャードのキースペースは、残りのシャード間で均等に分散されます。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group-shard-configuration \  
  --replication-group-id my-cluster \  
  --node-group-count 2 \  
  --node-groups-to-remove "0002" "0003" \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group-shard-configuration ^  
  --replication-group-id my-cluster ^  
  --node-group-count 2 ^  
  --node-groups-to-remove "0002" "0003" ^  
  --apply-immediately
```

シャードの削除 (ElastiCache API)

ElastiCache API を使用して、`ModifyReplicationGroupShardConfiguration` オペレーションを使用して Redis (クラスターモードが有効) クラスターのシャードをオンラインで再設定できます。

以下のプロセスでは、ElastiCache API を使用してシャードを削除することで、Redis (クラスターモードが有効) クラスター内のシャードを再設定する方法について説明します。

⚠ Important

レプリケーショングループからノードグループ (シャード) を削除する前に、はすべてのデータが残りのシャードに収まる ElastiCache ことを確認します。データが収まる場合、指定されたシャード (NodeGroupsToRemove) はリクエストに応じてレプリケーショングループから削除され、キースペースは残りのシャードにマッピングされます。データが残りのノードグループに収まらない場合、プロセスは終了し、レプリケーショングループはリクエスト前と同じノードグループ設定のままになります。

ElastiCache API を使用して、Redis (クラスターモードが有効) クラスターから 1 つ以上のシャードを削除できます。レプリケーショングループのすべてのシャードを削除することはできません。代わりに、レプリケーショングループを削除する必要があります。詳細については、「[レプリケーショングループの削除](#)」を参照してください。

ModifyReplicationGroupShardConfiguration を使って以下のパラメータを使用します。

パラメータ

- ApplyImmediately=true – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。
- ReplicationGroupId – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。
- NodeGroupCount – 必須。オペレーションの完了時に存在するシャード (ノードグループ) 数を指定します。シャードを削除する場合、NodeGroupCount の値は現在のシャード数より小さくなければなりません。
- NodeGroupsToRemove – --node-group-count が現在のノードグループ (シャード) 数より少ない場合は必須です。レプリケーショングループから削除するシャード (ノードグループ) ID の一覧。

次の手順では、1 つ以上のシャードを削除する手順を説明します。

Example - シャードの削除

次の例では、Redis (クラスターモードが有効) クラスター my-cluster から 2 つのノードグループを削除します。オペレーション完了時に合計 2 つのノードグループが存在することになります。削除されたシャードのキースペースは、残りのシャード間で均等に分散されます。

```
https://elasticache.us-east-2.amazonaws.com/  
?Action=ModifyReplicationGroupShardConfiguration  
&ApplyImmediately=true  
&NodeGroupCount=2  
&ReplicationGroupId=my-cluster  
&NodeGroupsToRemove.member.1=0002  
&NodeGroupsToRemove.member.2=0003  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20171002T192317Z  
&X-Amz-Credential=<credential>
```

オンラインのシャード再分散

、または ElastiCache API を使用して AWS Management Console AWS CLI、Redis (クラスターモードが有効) クラスター内のシャードを再調整できます。

トピック

- [オンラインのシャード再分散 \(コンソール\)](#)
- [オンラインのシャード再分散 \(AWS CLI\)](#)
- [オンラインシャードの再調整 \(ElastiCache API\)](#)

オンラインのシャード再分散 (コンソール)

以下のプロセスでは、AWS Management Consoleを使用してシャードを再分散し、Redis (クラスターモードが有効) クラスターでシャードの再構成を行う方法について説明します。

Redis (クラスターモードが有効) クラスターでシャード間のキースペースを再分散するには

1. <https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. 再分散する Redis (クラスターモードが有効) クラスターの名前 (名前の左にあるボックスではなく) を選択します。

i Tip

Redis (クラスターモードが有効) クラスターは、[シャード] 列で 1 より大きい値を持ちます。

- [再分散] を選択します。
- プロンプトが表示されたら、[再分散] を選択します。次のようなメッセージが表示されることがあります。#####Nothing to do#####: AmazonElasti#####: 400#####: InvalidReplicationGroupState##### ID: 2246cebd-9721-11e7-8d5b-e1b0f086c8cf)。この場合、[キャンセル] を選択します。

オンラインのシャード再分散 (AWS CLI)

modify-replication-group-shard-configuration を使って以下のパラメータを使用します。

パラメータ

- apply-immediately – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。
- replication-group-id – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。
- node-group-count – 必須。クラスター内のすべてのシャードでキースペースを再分散するため、この値は現在のシャード数と同じである必要があります。

以下のプロセスでは、AWS CLIを使用してシャードを再分散し、Redis (クラスターモードが有効) クラスターでシャードの再構成を行う方法について説明します。

Example - クラスターのシャードの再分散

次の例では、Redis (クラスターモードが有効) クラスター my-cluster のスロットを再分散して、スロットができるだけ等分に分散されるようにします。--node-group-count の値 (4) は、クラスターの現在のシャード数です。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group-shard-configuration \  
  --replication-group-id my-cluster \  
  --node-group-count 4
```

```
--node-group-count 4 \  
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group-shard-configuration ^  
  --replication-group-id my-cluster ^  
  --node-group-count 4 ^  
  --apply-immediately
```

オンラインシャードの再調整 (ElastiCache API)

ElastiCache API を使用して、ModifyReplicationGroupShardConfiguration オペレーションを使用して Redis (クラスターモードが有効) クラスターのシャードをオンラインで再設定できます。

ModifyReplicationGroupShardConfiguration を使って以下のパラメータを使用します。

パラメータ

- ApplyImmediately=true – 必須。シャードの再構成オペレーションがすぐに開始するよう指定します。
- ReplicationGroupId – 必須。シャードの再構成オペレーションをどのレプリケーショングループ (クラスター) で実行するかを指定します。
- NodeGroupCount – 必須。クラスター内のすべてのシャードでキースペースを再分散するため、この値は現在のシャード数と同じである必要があります。

以下のプロセスでは、ElastiCache API を使用してシャードを再調整することで、Redis (クラスターモードが有効) クラスター内のシャードを再設定する方法について説明します。

Example - クラスターの再分散

次の例では、Redis (クラスターモードが有効) クラスター `my-cluster` のスロットを再分散して、スロットができるだけ等分に分散されるようにします。NodeGroupCount の値 (4) は、クラスターの現在のシャード数です。

```
https://elasticache.us-east-2.amazonaws.com/  
?Action=ModifyReplicationGroupShardConfiguration  
&ApplyImmediately=true  
&NodeGroupCount=4
```

```
&ReplicationGroupId=my-cluster
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20171002T192317Z
&X-Amz-Credential=<credential>
```

ノードタイプの変更によるオンライン垂直スケーリング

Amazon ElastiCache for Redis バージョン 3.2.10 以降でオンライン垂直スケーリングを使用すると、最小限のダウンタイムで Redis クラスターを動的にスケーリングできます。これにより、Redis クラスターはスケーリング中であってもリクエストを処理できます。

Note

データ階層化を使用するクラスター (r6gd ノードタイプを使用するクラスターなど) と、データ階層化を使用しないクラスター (r6g ノードタイプを使用するクラスターなど) 間のスケーリングはサポートされていません。詳細については、「[データ階層化](#)」を参照してください。

以下の操作を行うことができます。

- [スケールアップ] – より大きいノードタイプを使用するように Redis クラスターのノードタイプを調整することで、読み取りおよび書き込み容量を増やします。

ElastiCache は、オンラインのままリクエストを処理しながら、クラスターのサイズを動的に変更します。

- [スケールダウン] – より小さいノードを使用するようにノードタイプを調整することで、読み取りおよび書き込み容量を減らします。スケールダウンする 繰り返しになりますが、オンラインのままリクエストを処理しながら、クラスターのサイズを ElastiCache 動的に変更します。この場合、ノードのサイズを小さくすることでコストを削減します。

Note

スケールアップおよびスケールダウンプロセスは、新しく選択されたノードタイプでクラスターを作成し、新しいノードを以前のノードと同期させることに依存します。スケールアップ/ダウンフローをスムーズにするには、以下の手順を実行します。

- 十分な ENI (Elastic Network Interface) 容量があることを確認します。スケールダウンの場合は、ノードを小さくすることで予想されるトラフィックを吸収するのに十分なメモリがあることを確認します。

メモリ管理のベストプラクティスについては、「[予約メモリの管理](#)」を参照してください。

- 垂直スケーリングプロセスは、完全にオンラインのままになるように設計されており、古いノードと新しいノードとの間でデータを同期させることに依存します。データトラフィックが最小になると予想される時間帯にスケールアップ/ダウンを開始することをお勧めします。
- 可能であれば、ステージング環境でのスケーリング中にアプリケーションの動作をテストします。

目次

- [オンラインスケールアップ](#)
 - [Redis キャッシュクラスターのスケールアップ \(コンソール\)](#)
 - [Redis キャッシュクラスターのスケールアップ \(AWS CLI\)](#)
 - [Redis キャッシュクラスターのスケールアップ \(ElastiCache API\)](#)
- [オンラインスケールダウン](#)
 - [Redis キャッシュクラスターのスケールダウン \(コンソール\)](#)
 - [単一ノード Redis キャッシュクラスターのスケールダウン \(AWS CLI\)](#)
 - [Redis キャッシュクラスターのスケールダウン \(ElastiCache API\)](#)

オンラインスケールアップ

トピック

- [Redis キャッシュクラスターのスケールアップ \(コンソール\)](#)
- [Redis キャッシュクラスターのスケールアップ \(AWS CLI\)](#)
- [Redis キャッシュクラスターのスケールアップ \(ElastiCache API\)](#)

Redis キャッシュクラスターのスケールアップ (コンソール)

次の手順では、ElastiCache マネジメントコンソールを使用して Redis クラスターをスケールアップする方法について説明します。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

Redis クラスターをスケールアップするには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターのリストから、クラスターを選択します。
4. [変更] を選択します。
5. [Modify Cluster] ウィザードで:
 - Node type リストから、スケールアップするノードタイプを選択します。スケールアップするには、既存のノードよりも大きいノードタイプを選択します。
6. スケールアッププロセスをすぐに実行する場合は、[すぐに適用] ボックスを選択します。[Apply immediately] ボックスを選択していない場合、スケールアッププロセスはこのクラスターの次のメンテナンス期間中に実行されます。
7. [変更] を選択します。

前の手順で [すぐに適用] を選択した場合、クラスターのステータスは [変更中] に変わります。ステータスが 使用可能 に変わると、変更は完了し、新しいクラスターの使用を開始できます。

Redis キャッシュクラスターのスケールアップ (AWS CLI)

以下の手順では、AWS CLIを使用して Redis キャッシュクラスターをスケールアップする方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

Redis キャッシュクラスターをスケールアップするには (AWS CLI)

1. 次のパラメータを指定して `list-allowed-node-type-modifications` コマンドを実行して、AWS CLI スケールアップできるノードタイプを決定します。

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --replication-group-id my-replication-group-id
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^  
  --replication-group-id my-replication-group-id
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ScaleUpModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",  
    "cache.m3.xlarge",  
    "cache.m4.10xlarge",  
    "cache.m4.2xlarge",  
    "cache.m4.4xlarge",  
    "cache.m4.large",  
    "cache.m4.xlarge",  
    "cache.r3.2xlarge",  
    "cache.r3.4xlarge",  
    "cache.r3.8xlarge",  
    "cache.r3.large",  
    "cache.r3.xlarge"  
  ],  
  "ScaleDownModifications": [  
    "cache.t2.micro",  
    "cache.t2.small",  
    "cache.t2.medium",  
    "cache.t1.small"  
  ],  
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

2. コマンドと以下のパラメータを使用して AWS CLI `modify-replication-group`、レプリケーショングループを変更して新しいより大きなノードタイプにスケールアップします。
 - `--replication-group-id` – スケールアップするレプリケーショングループの名前。

- `--cache-node-type` – キャッシュクラスターのスケーリング後の新しいノードタイプ。この値は、ステップ 1 で `list-allowed-node-type-modifications` コマンドによって返されるノードタイプのいずれかであることが必要です。
- `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できません。
- `--apply-immediately` – スケールアッププロセスがすぐに適用されるようにします。スケールアッププロセスをクラスターの次のメンテナンス期間に延期するには、`--no-apply-immediately` パラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id my-redis-cluster \  
  --cache-node-type cache.m3.xlarge \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id my-redis-cluster ^  
  --cache-node-type cache.m3.xlarge ^  
  --apply-immediately
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ReplicationGroup": {  
    "Status": "modifying",  
    "Description": "my-redis-cluster",  
    "NodeGroups": [  
      {  
        "Status": "modifying",  
        "Slots": "0-16383",  
        "NodeGroupId": "0001",
```

```
        "NodeGroupMembers": [
            {
                "PreferredAvailabilityZone": "us-east-1f",
                "CacheNodeId": "0001",
                "CacheClusterId": "my-redis-cluster-0001-001"
            },
            {
                "PreferredAvailabilityZone": "us-east-1d",
                "CacheNodeId": "0001",
                "CacheClusterId": "my-redis-cluster-0001-002"
            }
        ]
    },
    "ConfigurationEndpoint": {
        "Port": 6379,
        "Address": "my-redis-cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
    },
    "ClusterEnabled": true,
    "ReplicationGroupId": "my-redis-cluster",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "enabled",
    "SnapshotWindow": "07:30-08:30",
    "MemberClusters": [
        "my-redis-cluster-0001-001",
        "my-redis-cluster-0001-002"
    ],
    "CacheNodeType": "cache.m3.xlarge",
    "DataTiering": "disabled",
    "PendingModifiedValues": {}
}
}
```

詳細については、AWS CLI リファレンスの「[modify-replication-group](#)」を参照してください。

3. を使用した場合は`--apply-immediately`、次のパラメータを指定してコマンドを使用して AWS CLI `describe-cache-clusters` キャッシュクラスターのステータスを確認します。ステータスが `[available (使用可能)]` に変わると、新しいより大きいキャッシュクラスターノードの使用を開始できます。

Redis キャッシュクラスターのスケールアップ (ElastiCache API)

次のプロセスでは、ElastiCache API を使用してキャッシュクラスターを現在のノードタイプから新しいより大きなノードタイプにスケールアップします。このプロセス中、ElastiCache For Redis は DNS エントリを更新して、新しいノードをポイントします。このため、アプリケーションのエンドポイントを更新する必要はありません。Redis 5.0.5 以降では、クラスターがオンラインのままに受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケールアップできます。バージョン 4.0.10 以前では、DNS エントリが更新されている間、以前のバージョンのプライマリノードからの読み取りと書き込みが短時間中断することがあります。

より大きいノードタイプへのスケールアップにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

Redis キャッシュクラスターをスケールアップするには (ElastiCache API)

1. 次のパラメータを指定して、ElastiCache API `ListAllowedNodeTypeModifications` アクションを使用してスケールアップできるノードタイプを決定します。
 - `ReplicationGroupId` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListAllowedNodeTypeModifications  
&ReplicationGroupId=MyReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、「Amazon ElastiCache API リファレンス [ListAllowedNodeTypeModifications](#)」の「」を参照してください。

2. `ModifyReplicationGroup` ElastiCache API アクションと以下のパラメータを使用して、現在のレプリケーショングループを新しいノードタイプにスケールアップします。
 - `ReplicationGroupId` – レプリケーショングループの名前。

- `CacheNodeType` – このレプリケーショングループのキャッシュクラスターの新しいより大きいノードタイプ。この値は、手順 1 で `ListAllowedNodeTypeModifications` アクションによって返されるインスタンスタイプのいずれかであることが必要です。
- `CacheParameterGroupName` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
- `ApplyImmediately` – スケールアッププロセスがすぐに適用されるようにするには、`true` に設定します。スケールアッププロセスを次のメンテナンス期間に延期するには、`ApplyImmediately=false` を使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyReplicationGroup  
&ApplyImmediately=true  
&CacheNodeType=cache.m3.2xlarge  
&CacheParameterGroupName=redis32-m3-2x1  
&ReplicationGroupId=myReplGroup  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20141201T220302Z  
&Version=2014-12-01  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Date=20141201T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

詳細については、「Amazon ElastiCache API リファレンス [ModifyReplicationGroup](#)」の「」を参照してください。

3. を使用した場合は `ApplyImmediately=true`、以下のパラメータを指定して ElastiCache API `DescribeReplicationGroups` アクションを使用してレプリケーショングループのステータスをモニタリングします。ステータスが `[modifying]` から `[available]` に変わると、スケールアップした新しいレプリケーショングループへの書き込みを開始できます。
- `ReplicationGroupId` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeReplicationGroups  
&ReplicationGroupId=MyReplGroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

詳細については、「Amazon ElastiCache API リファレンス [DescribeReplicationGroups](#)」の「」を参照してください。

オンラインスケールダウン

トピック

- [Redis キャッシュクラスターのスケールダウン \(コンソール\)](#)
- [単一ノード Redis キャッシュクラスターのスケールダウン \(AWS CLI\)](#)
- [Redis キャッシュクラスターのスケールダウン \(ElastiCache API\)](#)

Redis キャッシュクラスターのスケールダウン (コンソール)

次の手順では、ElastiCache マネジメントコンソールを使用して Redis クラスターをスケールダウンする方法について説明します。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

Redis クラスターをスケールダウンするには (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターのリストから、希望するクラスターを選択します。
4. [変更] を選択します。
5. [Modify Cluster] ウィザードで:

- Node type リストから、スケーリングするノードタイプを選択します。スケールダウンするには、既存のノードより小さいノードタイプを選択します。すべてのノードタイプがスケールダウンできるわけではないことに注意してください。
6. スケールダウンプロセスをすぐに実行する場合は、[すぐに適用] ボックスを選択します。[すぐに適用] ボックスを選択していない場合、スケールダウンプロセスはこのクラスターの次のメンテナンスウィンドウ中に実行されます。
 7. [変更] を選択します。

前の手順で [すぐに適用] を選択した場合、クラスターのステータスは [変更中] に変わります。ステータスが 使用可能 に変わると、変更は完了し、新しいクラスターの使用を開始できます。

単一ノード Redis キャッシュクラスターのスケールダウン (AWS CLI)

以下の手順では、AWS CLIを使用して Redis キャッシュクラスターをスケールダウンする方法について説明しています。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

Redis キャッシュクラスターをスケールダウンするには (AWS CLI)

1. 次のパラメータを指定して `list-allowed-node-type-modifications` コマンドを実行して、AWS CLI スケールダウンできるノードタイプを決定します。

Linux、macOS、Unix の場合:

```
aws elasticache list-allowed-node-type-modifications \  
  --replication-group-id my-replication-group-id
```

Windows の場合:

```
aws elasticache list-allowed-node-type-modifications ^\  
  --replication-group-id my-replication-group-id
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ScaleUpModifications": [  
    "cache.m3.2xlarge",  
    "cache.m3.large",
```

```
    "cache.m3.xlarge",
    "cache.m4.10xlarge",
    "cache.m4.2xlarge",
    "cache.m4.4xlarge",
    "cache.m4.large",
    "cache.m4.xlarge",
    "cache.r3.2xlarge",
    "cache.r3.4xlarge",
    "cache.r3.8xlarge",
    "cache.r3.large",
    "cache.r3.xlarge"
  ]

  "ScaleDownModifications": [
    "cache.t2.micro",
    "cache.t2.small ",
    "cache.t2.medium ",
    "cache.t1.small"
  ]
}
```

詳細については、AWS CLI リファレンスの「[list-allowed-node-type-modifications](#)」を参照してください。

2. コマンドと以下のパラメータを使用して AWS CLI `modify-replication-group`、レプリケーショングループを変更して新しい小さなノードタイプにスケールダウンします。
 - `--replication-group-id` – スケールダウンするレプリケーショングループの名前。
 - `--cache-node-type` – キャッシュクラスターのスケールアップ後の新しいノードタイプ。この値は、ステップ 1 で `list-allowed-node-type-modifications` コマンドによって返されるノードタイプのいずれかであることが必要です。
 - `--cache-parameter-group-name` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
 - `--apply-immediately` – スケールアッププロセスがすぐに適用されるようにします。スケールダウンプロセスをクラスターの次のメンテナンスウィンドウまで延期するには、`--no-apply-immediately` パラメータを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id my-redis-cluster \  
  --cache-node-type cache.t2.micro \  
  --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id my-redis-cluster ^  
  --cache-node-type cache.t2.micro ^  
  --apply-immediately
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{  
  "ReplicationGroup": {  
    "Status": "modifying",  
    "Description": "my-redis-cluster",  
    "NodeGroups": [  
      {  
        "Status": "modifying",  
        "Slots": "0-16383",  
        "NodeGroupId": "0001",  
        "NodeGroupMembers": [  
          {  
            "PreferredAvailabilityZone": "us-east-1f",  
            "CacheNodeId": "0001",  
            "CacheClusterId": "my-redis-cluster-0001-001"  
          },  
          {  
            "PreferredAvailabilityZone": "us-east-1d",  
            "CacheNodeId": "0001",  
            "CacheClusterId": "my-redis-cluster-0001-002"  
          }  
        ]  
      }  
    ]  
  },  
  ],  
}
```



```
    "ConfigurationEndpoint": {
      "Port": 6379,
      "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
    },
    "ClusterEnabled": true,
    "ReplicationGroupId": "my-redis-cluster",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "enabled",
    "SnapshotWindow": "07:30-08:30",
    "MemberClusters": [
      "my-redis-cluster-0001-001",
      "my-redis-cluster-0001-002"
    ],
    "CacheNodeType": "cache.t2.micro",
    "DataTiering": "disabled"
    "PendingModifiedValues": {}
  }
}
```

詳細については、AWS CLI リファレンスの「[modify-replication-group](#)」を参照してください。

3. を使用した場合は--apply-immediately、次のパラメータを指定して コマンドを使用して AWS CLI describe-cache-clusters キャッシュクラスターのステータスを確認します。ステータスが [available (使用可能)] に変わると、新しいより小さいキャッシュクラスターノードの使用を開始できます。

Redis キャッシュクラスターのスケールダウン (ElastiCache API)

次のプロセスでは、ElastiCache API を使用してレプリケーショングループを現在のノードタイプから新しい小さいノードタイプにスケールリングします。このプロセス中、Redis クラスターは最小限のダウンタイムでリクエストを処理し続けます。

より小さいノードタイプへのスケールダウンにかかる時間はノードタイプと現在のキャッシュクラスターのデータ量によって異なります。

スケールダウン (ElastiCache API)

1. 次のパラメータを指定して ElastiCache API ListAllowedNodeTypeModifications アクションを使用して、スケールダウンできるノードタイプを決定します。

- `ReplicationGroupId` – レプリケーショングループの名前。すべてのレプリケーショングループではなく特定のレプリケーショングループの定義を表示するには、このパラメータを使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=ListAllowedNodeTypeModifications  
  &ReplicationGroupId=MyReplGroup  
  &Version=2015-02-02  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20150202T192317Z  
  &X-Amz-Credential=<credential>
```

詳細については、「Amazon ElastiCache API リファレンス [ListAllowedNodeTypeModifications](#)」の「」を参照してください。

2. `ModifyReplicationGroup` ElastiCache API アクションと以下のパラメータを使用して、現在のレプリケーショングループを新しいノードタイプにスケールダウンします。

- `ReplicationGroupId` – レプリケーショングループの名前。
- `CacheNodeType` – このレプリケーショングループのキャッシュクラスターの新しいより小さいノードタイプ。この値は、手順 1 で `ListAllowedNodeTypeModifications` アクションによって返されるインスタンスタイプのいずれかであることが必要です。
- `CacheParameterGroupName` – (オプション) `reserved-memory` を使用してクラスターの予約メモリを管理する場合は、このパラメータを使用します。新しいノードタイプ用の適切な容量のメモリを確保するカスタムキャッシュパラメータグループを指定します。`reserved-memory-percent` を使用している場合は、このパラメータを省略できます。
- `ApplyImmediately` – スケールダウンプロセスがすぐに適用されるようにするには、`true` に設定します。スケールダウンプロセスを次のメンテナンスウィンドウに延期するには、`ApplyImmediately=false` を使用します。

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=ModifyReplicationGroup  
  &ApplyImmediately=true  
  &CacheNodeType=cache.t2.micro  
  &CacheParameterGroupName=redis32-m3-2x1  
  &ReplicationGroupId=myReplGroup
```

```
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&Version=2014-12-01
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

詳細については、「Amazon ElastiCache API リファレンス [ModifyReplicationGroup](#)」の「」を参照してください。

ElastiCache for Redis での JSON の使用開始

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

JSON 上で動作するアプリケーションに対して Redis API 操作を使用することに加えて、オブジェクト全体を操作しなくても、JSON ドキュメントの特定の部分を効率的に取得および更新できるようになりました。これにより、パフォーマンスを向上させ、コストを削減できます。また、[Goessner-style](#) JSONPath クエリを使用して、JSON ドキュメントの内容を検索することもできます。

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

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

Note

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

トピック

- [Redis JSON データ型の概要](#)
- [サポートされている Redis JSON コマンド](#)

Redis JSON データ型の概要

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

用語

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

サポートされている JSON 標準

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

ルート要素

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

ドキュメントサイズの制限

JSON ドキュメントは、迅速なアクセスおよび変更のために最適化された形式で内部的に格納されます。通常、この形式では、同じドキュメントのシリアル化された同等の表現よりもいくらか多くのメモリを消費することになります。

単一の JSON ドキュメントによるメモリ消費量は 64 MB に制限されています。これは JSON 文字列ではなく、インメモリデータ構造のサイズです。JSON.DEBUG MEMORY コマンドを使用すれば、JSON ドキュメントが消費するメモリの量を確認できます。

JSON ACLs

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

ACL

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

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

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

ネスト深度の制限

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

コマンド構文

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

表記法:

- 必須引数は山括弧で囲みます。例: <key>
- オプションの引数は角括弧で囲みます。例: [path]
- 追加のオプション引数は省略記号 (「...」) で示されます。例: [json ...]

パス構文

Redis JSON では、次の 2 種類のパス構文をサポートしています。

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

Note

一部のコマンドの結果は、使用されるパス構文のタイプの影響を受けます。

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

拡張構文

記号/式	説明
\$	ルート要素。
. または []	子演算子。
..	再帰下降。
*	ワイルドカード。オブジェクトまたは配列のすべての要素。
[]	配列の添字演算子。インデックスは 0 ベースです。
[.]	union 演算子。
[start:end:step]	配列のスライス演算子。
?()	フィルタ (スクリプト) 式を現在の配列またはオブジェクトに適用します。

記号/式	説明
()	フィルタ式。
@	処理中の現在のノードを参照するフィルタ式で使用されます。
==	等しい。フィルタ式で使用されます。
!=	等しくない。フィルタ式で使用されます。
>	より大きい。フィルタ式で使用されます。
>=	以上。フィルタ式で使用されます。
<	より小さい。フィルタ式で使用されます。
<=	以下。フィルタ式で使用されます。
&&	論理 AND。複数のフィルタ式を組み合わせるために使用されます。
	論理 OR。複数のフィルタ式を組み合わせるために使用されます。

例

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

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

```

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

```

パス	説明
<code>\$.store.book[*].author</code>	この店のすべての本の著者です。
<code>\$..author</code>	すべての著者です。
<code>\$.store.*</code>	店のすべてのメンバー。
<code>\$["store"].*</code>	店のすべてのメンバー。
<code>\$.store..price</code>	店のすべてのものの価格です。

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

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

追加のフィルタ式の例:

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

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


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

制限構文

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

例

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

 Note

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

一般的なエラープレフィックス

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

プレフィックス	説明
ERR	一般的なエラーです。

プレフィックス	説明
LIMIT	サイズ制限を超えたときに発生するエラーです。例えば、ドキュメントのサイズ制限やネストの深度制限を超えた場合などです。
NONEXISTENT	キーまたはパスが存在しません。
OUTOFBOUNDARIES	配列インデックスが範囲外です。
SYNTAXERR	構文エラーです。
WRONGTYPE	値のタイプが間違っています。

JSON 関連メトリクス

以下の JSON 情報メトリクスが提供されます。

情報	説明
json_total_memory_bytes	JSON オブジェクトに割り当てられたメモリの合計です。
json_num_documents	Redis 内のドキュメントの総数です。

コアメトリクスを照会するには、以下の Redis コマンドを実行します。

```
info json_core_metrics
```

ElastiCache for Redis と JSON の連携方法

以下のセクションでは、ElastiCache for Redis と JSON データ型の連携方法について説明します。

演算子の優先順位

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

最大パスネスト制限の動作

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

数値の処理

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

数値表現:

入力で JSON 数値を受け取ると、次の 2 つの内部バイナリ表現のいずれかに変換されます: 64 ビット符号付き整数、または 64 ビット IEEE 倍精度浮動小数点数。元の文字列、およびそのすべての書式は保持されません。そのため、数値が JSON 応答の一部として出力されるときに、内部のバイナリ表現が、一般的な書式ルールが使用された印刷可能文字列に変換されます。これらのルールにより、受信した文字列とは異なる文字列が生成される場合があります。

算術コマンド NUMINCRBY および NUMMULTBY:

- 両方の数値が整数で、結果が int64 の範囲外である場合は、自動的に 64 ビット IEEE 倍精度浮動小数点数になります。
- 数字の少なくとも 1 つが浮動小数点の場合、結果は 64 ビット IEEE 倍精度浮動小数点数になります。
- 結果が 64 ビット IEEE 倍精度の範囲を超える場合は、OVERFLOW エラーが返されます。

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

直接配列フィルタリング

ElastiCache for Redis では、配列オブジェクトを直接フィルタリングします。

[0,1,2,3,4,5,6] のようなデータと \$[?(@<4)] のようなパスクエリ、あるいは {"my_key": [0,1,2,3,4,5,6]} のようなデータと \$.my_key[?(@<4)] のようなパスクエリの場合、ElastiCache for Redis はどちらの状況でも [1,2,3] を返します。

配列インデックス作成の動作

ElastiCache for Redis では、配列で正と負の両方のインデックスを使用できます。長さ 5 の配列の場合、0 で最初の要素を照会し、1 で 2 番目の要素を照会する、という順序になります。負の数は配列

の最後から始まるので、-1 は 5 番目の要素を照会し、-2 は 4 番目の要素を照会し、以下同様に続きます。

お客様の予測可能な動作を保証するために、ElastiCache for Redis では配列インデックスの切り捨て/切り上げを行いません。したがって、長さ 5 の配列がある場合、インデックス 5 以上、または -6 以下を呼び出しても結果は生成されません。

厳密な構文評価

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

サポートされている Redis JSON コマンド

ElastiCache for Redis は、以下の Redis JSON コマンドをサポートしています。

トピック

- [JSON.ARRAPPEND](#)
- [JSON.ARRINDEX](#)
- [JSON.ARRINSERT](#)
- [JSON.ARRLEN](#)
- [JSON.ARRPOP](#)
- [JSON.ARRTRIM](#)
- [JSON.CLEAR](#)
- [JSON.DEBUG](#)
- [JSON.DEL](#)
- [JSON.FORGET](#)
- [JSON.GET](#)
- [JSON.MGET](#)
- [JSON.NUMINCRBY](#)
- [JSON.NUMMULTBY](#)
- [JSON.OBJLEN](#)
- [JSON.OBJKEYS](#)
- [JSON.RESP](#)

- [JSON.SET](#)
- [JSON.STRAPPEND](#)
- [JSON.STRLEN](#)
- [JSON.TOGGLE](#)
- [JSON.TYPE](#)

JSON.ARRAPPEND

パスの配列値に 1 つ以上の値を追加します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.ARRINDEX

パスの配列で最初に出現するスカラー JSON 値を検索します。

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

構文

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

- key (必須) - JSON ドキュメントタイプの Redis キー。
- path (必須) - JSON パス。
- json-scalar (必須) - 検索するスカラー値。JSON スカラーは、オブジェクトでも配列でもない値を指します。すなわち、文字列、数値、ブール、null がスカラー値です。

- `start` (オプション) – 開始インデックス (この値を含みます)。指定しない場合、デフォルトで 0 になります。
- `end` (オプション) – 終了インデックス (この値を含みません)。指定しない場合、デフォルトで 0 になります。したがって、最後の要素が含まれます。0 または -1 は、最後の要素が含まれることを意味します。

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.ARRINSERT

そのインデックスの前のパスの配列値に 1 つ以上の値を挿入します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]'  
OK  
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
```

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

制限パス構文:

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

JSON.ARRLEN

パスの配列値の長さを取得します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

- 一括文字列の配列。各要素はオブジェクトのキー名です。

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

例

拡張パス構文:

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

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

制限パス構文:

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

127.0.0.1:6379> JSON.SET k2 . '[[[], \"a\", [\"a\", \"b\"], [\"a\", \"b\", \"c\"], 4]]'
OK
```

```
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
1) (integer) 2
```

JSON.ARRPOP

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

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

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

```
127.0.0.1:6379> JSON.GET k2  
"[[\"a\"],[\"a\"],[\"b\"]]"
```

JSON.ARRTRIM

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

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

構文


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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.CLEAR

パスの配列またはオブジェクトをクリアします。

構文

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

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

戻る

- 整数、クリアされたコンテナの数。
- 空の配列またはオブジェクトをクリアすると、1つのコンテナがクリアされます。
- コンテナ以外の値をクリアすると 0 が返されます。

例

```
127.0.0.1:6379> JSON.SET k1 . '[[[], [0], [0,1], [0,1,2], 1, true, null, "d"]]'
OK
127.0.0.1:6379> JSON.CLEAR k1 $[*]
(integer) 7
127.0.0.1:6379> JSON.CLEAR k1 $[*]
(integer) 4
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
```

```
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"
```

JSON.DEBUG

レポート情報。サポートされるサブコマンドは以下のとおりです。

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

構文

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

サブコマンドによって異なります。

MEMORY

- パスが拡張構文の場合:
 - 各パスにおける JSON 値のメモリサイズ (バイト単位) を表す整数の配列を返します。
 - Redis キーが存在しない場合は、空の配列を返します。
- パスが制限構文の場合:
 - 整数のメモリサイズ、および JSON 値 (バイト単位) を返します。
 - Redis キーが存在しない場合は、null を返します。

FIELDS

- パスが拡張構文の場合:

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

HELP - ヘルプメッセージの配列を返します。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2}, [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
```

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

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

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

JSON.DEL

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

構文

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

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

戻る

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.FORGET

[JSON.DEL](#) のエイリアス。

JSON.GET

1 つ以上のパスにあるシリアル化された JSON を返します。

構文

```
JSON.GET <key>
```

```
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

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

戻る

拡張パス構文:

パスが 1 つ指定されている場合:

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

複数のパスが指定されている場合:

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

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
```

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

制限パス構文:

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

JSON.MGET

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

構文

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


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

戻る

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

例

拡張パス構文:

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

制限パス構文:

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

```
OK
```

```
127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
```

```
1) "\"New York\""
```

```
2) "\"Seattle\""
```

```
3) "\"Seattle\""
```

JSON.NUMINCRBY

指定された数だけパスの数値を増分します。

構文

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

- **key (必須)** - JSON ドキュメントタイプの Redis キー。
- **path (必須)** - JSON パス。
- **number (必須)** — 数値。

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

例

拡張パス構文:

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

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

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

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a", "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
```

```

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

```

制限パス構文:

```

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

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

127.0.0.1:6379> JSON.SET k2 . '{"a:{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1

```

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

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
  "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
"4"
```

JSON.NUMMULTBY

指定された数でパスの数値を乗算します。

構文

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

- **key (必須)** - JSON ドキュメントタイプの Redis キー。
- **path (必須)** - JSON パス。
- **number (必須)** — 数値。

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

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

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

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

制限パス構文:

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

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

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

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

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


JSON.OBJLEN

パスにあるオブジェクト値のキーの数を取得します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

```
127.0.0.1:6379> JSON.OBJLEN k1 $.a
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)
```

制限パス構文:

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

```
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

JSON.OBJKEYS

パスにあるオブジェクト値のキー名を取得します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.RESP

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

- JSON null は、RESP Null 一括文字列にマップされます。
- JSON ブール値は、それぞれの RESP 単純文字列にマッピングされます。

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

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

OK

```
127.0.0.1:6379> JSON.RESP k1 $.address
```

```
1) 1) {  
  2) 1) "street"  
     2) "21 2nd Street"  
  3) 1) "city"  
     2) "New York"  
  4) 1) "state"  
     2) "NY"  
  5) 1) "zipcode"  
     2) "10021-3100"
```

```
127.0.0.1:6379> JSON.RESP k1 $.address.*
```

```
1) "21 2nd Street"  
2) "New York"  
3) "NY"  
4) "10021-3100"
```

```
127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
```

```
1) 1) [  
  2) 1) {  
     2) 1) "type"  
        2) "home"  
     3) 1) "number"  
        2) "555 555-1234"  
  3) 1) {  
     2) 1) "type"  
        2) "office"  
     3) 1) "number"  
        2) "555 555-4567"
```

```
127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
```

```
1) 1) {  
  2) 1) "type"  
     2) "home"  
  3) 1) "number"  
     2) "212 555-1234"  
2) 1) {  
  2) 1) "type"  
     2) "office"  
  3) 1) "number"  
     2) "555 555-4567"
```

制限パス構文:

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

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

127.0.0.1:6379> JSON.RESP k1
1) {
2) 1) "firstName"
   2) "John"
3) 1) "lastName"
   2) "Smith"
4) 1) "age"
   2) (integer) 27
5) 1) "weight"
   2) "135.25"
6) 1) "isAlive"
   2) true
7) 1) "address"
   2) 1) {
      2) 1) "street"
         2) "21 2nd Street"
      3) 1) "city"
         2) "New York"
      4) 1) "state"
         2) "NY"
```

```
5) 1) "zipcode"
    2) "10021-3100"
8) 1) "phoneNumbers"
    2) 1) [
        2) 1) {
            2) 1) "type"
            2) "home"
            3) 1) "number"
            2) "212 555-1234"
        3) 1) {
            2) 1) "type"
            2) "office"
            3) 1) "number"
            2) "555 555-4567"
    9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

JSON.SET

パスに JSON 値を設定します。

パスがオブジェクトメンバーを要求する場合:

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

パスが配列インデックスを要求する場合:

- 親要素が存在しない場合、このコマンドは NONEXISTENT エラーを返します。
- 親要素は存在するが配列ではない場合、このコマンドは ERROR を返します。
- 親要素は存在するが、インデックスが範囲外である場合、このコマンドは OUTFOFBOUNDARIES エラーを返します。

- 親要素が存在し、インデックスが有効な場合、要素は新しい JSON 値に置き換えられます。

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

構文

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

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

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

戻る

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

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'  
OK  
127.0.0.1:6379> JSON.SET k1 $.a.* '0'  
OK  
127.0.0.1:6379> JSON.GET k1  
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"  
  
127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
```

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

制限パス構文:

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

JSON.STRAPPEND

パスの JSON 文字列に文字列を追加します。

構文

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

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

戻る

パスが拡張構文の場合:

- 各パスの文字列の新しい長さを表す整数の配列。

- パスの値が文字列でない場合、対応する戻り値は null です。
- 入力された json 引数が有効な JSON 文字列でない場合は、SYNTAXERR エラーになります。
- パスが存在しない場合は、NONEXISTENT エラーになります。

パスが制限構文の場合:

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

例

拡張パス構文:

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

制限パス構文:

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

JSON.STRLLEN

パスの JSON 文字列値の長さを取得します。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

- 整数、文字列の長さ。

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

例

拡張パス構文:

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

制限パス構文:

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

```
(integer) 1
```

JSON.TOGGLE

パスのブール値を true と false の間で切り替えます。

構文

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

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

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

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

```
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

制限パス構文:

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

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

JSON.TYPE

指定されたパスの値の型を報告します。

構文

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

- **key (必須)** - JSON ドキュメントタイプの Redis キー。
- **path (オプション)** - JSON パス。指定しない場合、デフォルトでルートになります。

戻る

パスが拡張構文の場合:

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

パスが制限構文の場合:

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

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
OK
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
4) boolean
5) null
6) object
7) array
```

制限パス構文:

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



```
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

ElastiCache リソースのタグ付け

クラスターと他の ElastiCache リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。タグを使用すると、例えば用途別、所有者別、環境別などのさまざまな方法で AWS リソースを分類できます。これは、同じタイプのリソースが多数ある場合に役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。このトピックでは、タグとその作成方法について説明します。

Warning

ベストプラクティスとして、機密データをタグに含めないようお勧めします。

タグの基本

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

各リソースタイプのニーズを満たす一連のタグキーを考案することをお勧めします。一貫性のある一連のタグキーを使用することで、リソースの管理が容易になります。追加したタグに基づいてリソー

スを検索およびフィルタリングできます。効果的なリソースのタグ付け戦略を実装する方法の詳細については、「[AWS ホワイトペーパーのタグ付けのベストプラクティス](#)」を参照してください。

タグには、ElastiCache に関連する意味はなく、完全に文字列として解釈されます。また、タグは自動的にリソースに割り当てられます。タグのキーと値は編集でき、タグはリソースからいつでも削除できます。タグの値は null に設定できます。特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。リソースを削除すると、リソースのタグも削除されます。さらに、レプリケーショングループでタグを追加または削除すると、そのレプリケーショングループ内のすべてのノードにもタグが追加または削除されます。

AWS Management Console、AWS CLI、および ElastiCache API を使用してタグを操作できます。

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

タグを付けることができるリソース

アカウントにすでに存在するほとんどの ElastiCache リソースにタグ付けできます。以下の表に、タグ付けをサポートするリソースを示します。AWS Management Console を使用している場合、リソースにタグを適用するには、[タグエディタ](#)を使用します。一部のリソースの画面では、リソースの作成時にリソースのタグを指定できます。たとえば、Name のキーと指定した値をタグ付けします。ほとんどの場合、リソースの作成後すぐに (リソースの作成時ではなく) コンソールによりタグが適用されます。コンソールではリソースを [Name] タグに応じて整理できますが、このタグには ElastiCache サービスに対する意味論的意味はありません。

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

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

- CreateServerlessCache
- CreateCacheCluster
- CreateReplicationGroup


- CopyServerlessCacheSnapshot
- CopySnapshot
- CreateCacheParameterGroup
- CreateCacheSecurityGroup
- CreateCacheSubnetGroup
- CreateServerlessCacheSnapshot
- CreateSnapshot
- CreateUserGroup
- CreateUser
- PurchaseReservedCacheNodesOffering

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

ElastiCache リソースのタグ付けのサポート

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

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

 Note

グローバルデータストアにタグを付けることはできません。

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

詳細については、「[リソースのタグ付けの例](#)」を参照してください。

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

キャッシュとスナップショットのタグ付け

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

- **CreateReplicationGroup:**

- `--primary-cluster-id` および `--tags` パラメータがリクエストに含まれている場合、リクエストタグはレプリケーショングループに追加され、レプリケーショングループ内のすべてのキャッシュクラスターに伝播されます。プライマリキャッシュクラスターに既存のタグがある場合、これらはリクエストタグで上書きされ、すべてのノードで一貫したタグを持つようになります。

リクエストタグがない場合、プライマリキャッシュクラスタータグはレプリケーショングループに追加され、すべてのキャッシュクラスターに伝播されます。

- `--snapshot-name` または `--serverless-cache-snapshot-name` が供給された場合:

タグがリクエストに含まれている場合、レプリケーショングループはそれらのタグのみでタグ付けされます。タグがリクエストに含まれていない場合、スナップショットタグがレプリケーショングループに追加されます。

- `--global-replication-group-id` が供給された場合:

タグがリクエストに含まれている場合、リクエストタグはレプリケーショングループに追加され、すべてのキャッシュクラスターに伝播されます。

- **CreateCacheCluster:**

- `--replication-group-id` が供給された場合:

タグがリクエストに含まれている場合、キャッシュクラスターはそれらのタグのみでタグ付けされます。タグがリクエストに含まれていない場合、キャッシュクラスターはプライマリキャッシュクラスターのタグではなく、レプリケーショングループタグを継承します。

- `--snapshot-name` が供給された場合:

タグがリクエストに含まれている場合、キャッシュクラスターはそれらのタグのみでタグ付けされます。タグがリクエストに含まれていない場合、スナップショットタグはキャッシュクラスターに追加されます。

- **CreateServerlessCache:**

- タグがリクエストに含まれている場合、リクエストタグのみがサーバーレスキャッシュに追加されます。
- CreateSnapshot:
 - --replication-group-id が供給された場合:

タグがリクエストに含まれている場合、リクエストタグのみがスナップショットに追加されます。タグがリクエストに含まれていない場合、レプリケーショングループタグがスナップショットに追加されます。
 - --cache-cluster-id が供給された場合:

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

タグは、レプリケーショングループタグから伝播されます。
- CreateServerlessCacheSnapshot:
 - タグがリクエストに含まれている場合、リクエストタグのみがサーバーレスキャッシュのスナップショットに追加されます。
- CopySnapshot:
 - タグがリクエストに含まれている場合、リクエストタグのみがスナップショットに追加されます。タグがリクエストに含まれていない場合、コピー元のスナップショットタグがコピーされたスナップショットに追加されます。
- CopyServerlessCacheSnapshot:
 - タグがリクエストに含まれている場合、リクエストタグのみがサーバーレスキャッシュのスナップショットに追加されます。
- AddTagsToResource および RemoveTagsFromResource:
 - タグはレプリケーショングループに追加または削除され、アクションはレプリケーショングループ内のすべてのクラスターに伝播されます。

Note

AddTagsToResource および RemoveTagsFromResource は、デフォルトのパラメータおよびセキュリティグループには使用できません。

- IncreaseReplicaCount および ModifyReplicationGroupShardConfiguration:
 - レプリケーショングループに追加されたすべての新しいクラスターには、レプリケーショングループと同じタグが適用されます。

タグの制限

タグには以下のような基本制限があります。

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

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

タグ付けできる ElastiCache リソースの詳細については、「[タグを付けることができるリソース](#)」を参照してください。

リソースのタグ付けの例

- タグを使ったサーバーレスキャッシュを作成します。

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name CacheName \  
  --engine redis
```

```
--tags Key="Cost Center", Value="1110001" Key="project",Value="XYZ"
```

- サーバーレスキャッシュへのタグの追加

```
aws elasticache add-tags-to-resource \  
--resource-name arn:aws:elasticache:us-east-1:111111222233:serverlesscache:my-cache \  
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- レプリケーショングループにタグを追加します。

```
aws elasticache add-tags-to-resource \  
--resource-name arn:aws:elasticache:us-east-1:111111222233:replicationgroup:my-rg \  
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- タグを使用したキャッシュクラスターの作成。

```
aws elasticache create-cache-cluster \  
--cluster-id testing-tags \  
--cluster-description cluster-test \  
--cache-subnet-group-name test \  
--cache-node-type cache.t2.micro \  
--engine redis \  
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- タグ付きのサーバーレススナップショットの作成

```
aws elasticache create-serverless-cache-snapshot \  
--serverless-cache-name testing-tags \  
--serverless-cache-snapshot-name bkp-testing-tags-scs \  
--tags Key="work",Value="foo"
```

- タグ付きのスナップショットを作成します。

この場合、リクエストでタグを追加すると、レプリケーショングループにタグが含まれている場合でも、スナップショットはリクエストタグのみを受け取ります。

```
aws elasticache create-snapshot \  
--replication-group-id testing-tags \  
--snapshot-name bkp-testing-tags-rg \  
--tags Key="work",Value="foo"
```


タグベースのアクセスコントロールポリシーの例

1. クラスターに Project=XYZ というタグがある場合にのみ、クラスターへの AddTagsToResource アクションが許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "elasticache:AddTagsToResource",
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "XYZ"
        }
      }
    }
  ]
}
```

2. レプリケーショングループに Project タグと Service タグが含まれ、キーが Project と Service と異なる場合、レプリケーショングループからの RemoveTagsFromResource アクションが許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "elasticache:RemoveTagsFromResource",
      "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Service": "Elasticache",
          "aws:ResourceTag/Project": "XYZ"
        },
        "ForAnyValue:StringNotEqualsIgnoreCase": {

```

```

        "aws:TagKeys": [
            "Project",
            "Service"
        ]
    }
}
]
}

```

3. タグが Project と Service と異なる場合にのみ、任意のリソースへの AddTagsToResource が許可されます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "elasticache:AddTagsToResource",
      "Resource": [
        "arn:aws:elasticache:*:*:*:*"
      ],
      "Condition": {
        "ForAnyValue:StringNotEqualsIgnoreCase": {
          "aws:TagKeys": [
            "Service",
            "Project"
          ]
        }
      }
    }
  ]
}

```

4. リクエストに Tag Project=Foo がある場合、CreateReplicationGroup アクションが拒否されます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "elasticache:CreateReplicationGroup",

```

```

    "Resource": [
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/Project": "Foo"
      }
    }
  }
]
}

```

5. ソーススナップショットに Project=XYZ タグがあり、リクエストタグが Service=Elasticache の場合、CopySnapshot アクションが拒否されます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "elasticache:CopySnapshot",
      "Resource": [
        "arn:aws:elasticache:*:*:snapshot:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "XYZ",
          "aws:RequestTag/Service": "Elasticache"
        }
      }
    }
  ]
}

```

6. リクエストタグ Project が欠落しているか、Dev、QA、または Prod と等しくない場合、CreateCacheCluster アクションが拒否されます。

```

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

```

```

        "elasticache:CreateCacheCluster"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*",
        "arn:aws:elasticache:*:*:securitygroup:*",
        "arn:aws:elasticache:*:*:replicationgroup:*"
    ]
},
{
    "Effect": "Deny",
    "Action": [
        "elasticache:CreateCacheCluster"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
    ],
    "Condition": {
        "Null": {
            "aws:RequestTag/Project": "true"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:AddTagsToResource"
    ],
    "Resource": "arn:aws:elasticache:*:*:cluster:*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Project": [
                "Dev",
                "Prod",
                "QA"
            ]
        }
    }
}
]
}

```

条件キーの詳細については、「[条件キーの使用](#)」を参照してください。

コスト配分タグによるコストのモニタリング

Amazon ElastiCache でリソースにコスト配分タグを追加する場合、リソースのタグ値に基づいて請求書の費用をグループ化してコストを追跡できます。

ElastiCache コスト配分タグは、ElastiCache リソースを定義してそのリソースに関連付けるキーと値のペアです。キーと値は大文字と小文字が区別されます。タグキーを使用してカテゴリを定義し、タグ値をそのカテゴリの項目にすることができます。たとえば、「CostCenter」というタグキーと「10010」というタグ値を定義して、リソースがコストセンター 10010 に割り当てられていることを示すことができます。また、Environment などのキーと、test や production などの値を使用して、リソースがテスト用なのか本稼働用なのかを示すこともできます。リソースに関連付けられているコストの追跡が簡単になるように、一貫した一連のタグキーを使用することをお勧めします。

コスト配分タグを使用して AWS の請求書を分類し、自分のコスト構造を反映させます。そのためには、サインアップして、タグキー値が含まれた AWS アカウントの請求書を取得する必要があります。次に、結合したリソースのコストを見るには、同じタグキー値のリソースに従って請求書情報を整理します。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。

タグを組み合わせてさらに細かくコストを追跡することもできます。たとえば、リージョンごとのサービスのコストを追跡するために、Service と Region というタグキーを使用できます。1 つのリソースでは値を ElastiCache と Asia Pacific (Singapore) にし、別のリソースでは値を ElastiCache と Europe (Frankfurt) にします。これによって、ElastiCache の合計コストをリージョンごとに表示できます。詳細については、「[AWS Billing ユーザーガイド](#)」の「コスト配分タグの使用」(Use Cost Allocation Tags) を参照してください。

Redis ノードに ElastiCache コスト配分タグを追加できます。タグの追加やリスト、変更、削除を行った場合、そのオペレーションは、指定したノードにのみ適用されます。

ElastiCache コスト配分タグの特徴

- コスト配分タグは、ARN として CLI および API オペレーションで指定された ElastiCache リソースに適用されます。resource-type は "cluster" です。

サンプル ARN: `arn:aws:elasticache:<region>:<customer-id>:<resource-type>:<resource-name>`

サンプル `arn:arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

- タグキーは、必須のタグ名です。キーの文字列値は、長さが 1~128 文字の Unicode 文字です。aws: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、下線 (_)、ピリオド (.)、コロン (:)、バックスラッシュ (\)、等号 (=)、プラス記号 (+)、ハイフン (-)、またはアットマーク (@) を含めることができます。
- タグ値は、オプションのタグの値です。値の文字列値は、長さが 1~256 文字の Unicode 文字です。aws: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、下線 (_)、ピリオド (.)、コロン (:)、バックスラッシュ (\)、等号 (=)、プラス記号 (+)、ハイフン (-)、またはアットマーク (@) を含めることができます。
- ElastiCache リソースには、最大 50 個のタグを設定できます。
- 値はタグセット内で一意である必要はありません。たとえば、タグセット内に Service と Application というキーがあり、両方の値として ElastiCache を指定できます。

AWS はタグに意味論的意味を適用しません。タグは文字列として厳密に解釈されます。AWS によって ElastiCache リソースのタグは自動的に設定されません。

AWS CLI を使用したコスト配分タグの管理

AWS CLI を使用して、コスト配分タグを追加、変更、または削除できます。

サンプル `arn:arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

コスト配分タグは、ElastiCache for Redis ノードに適用されます。タグ付けされるノードは、ARN (Amazon リソースネーム) を使用して指定されます。

サンプル `arn:arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

トピック

- [AWS CLI を使用したタグの一覧表示](#)
- [AWS CLI を使用したタグの追加](#)
- [AWS CLI を使用したタグの変更](#)

- [AWS CLI を使用したタグの削除](#)

AWS CLI を使用したタグの一覧表示

[list-tags-for-resource](#) オペレーションを行い、AWS CLI を使用して既存の ElastiCache リソースのタグを一覧表示できます。

次のコードは、AWS CLI を使用して、us-west-2 リージョンの my-cluster クラスター Redis ノード my-cluster-001 のタグをリスト表示します。

Linux、macOS、Unix の場合:

```
aws elasticache list-tags-for-resource \  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

Windows の場合:

```
aws elasticache list-tags-for-resource ^  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

このオペレーションの出力は、リソースのすべてのタグを示した次のリストのようになります。

```
{  
  "TagList": [  
    {  
      "Value": "10110",  
      "Key": "CostCenter"  
    },  
    {  
      "Value": "EC2",  
      "Key": "Service"  
    }  
  ]  
}
```

リソースにタグが見つからない場合は、空の TagList が出力されます。

```
{  
  "TagList": []  
}
```

```
}
```

詳細については、「AWS CLI for ElastiCache [list-tags-for-resource](#)」を参照してください。

AWS CLI を使用したタグの追加

[add-tags-to-resource](#) CLI オペレーションを行い、AWS CLI を使用して、既存の ElastiCache リソースにタグを追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されます。

次のコードは、AWS CLI を使用して、us-west-2 リージョンの クラスター Service のノード Region に、elasticache と us-west-2 というキーを追加し、それぞれの値を my-cluster-001 と my-cluster に設定します。

Linux、macOS、Unix の場合:

```
aws elasticache add-tags-to-resource \  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \  
  --tags Key=Service,Value=elasticache \  
         Key=Region,Value=us-west-2
```

Windows の場合:

```
aws elasticache add-tags-to-resource ^  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^  
  --tags Key=Service,Value=elasticache ^  
         Key=Region,Value=us-west-2
```

このオペレーションの出力は、次のオペレーションのリソースのすべてのタグを示した以下のリストのようになります。

```
{  
  "TagList": [  
    {  
      "Value": "elasticache",  
      "Key": "Service"  
    },  
    {  
      "Value": "us-west-2",  
      "Key": "Region"  
    }  
  ]  
}
```



```
]
}
```

詳細については、「AWS CLI for ElastiCache [add-tags-to-resource](#)」を参照してください。

オペレーション [create-cache-cluster](#) を使用して新しいクラスターを作成するときに、AWS CLI を使用してクラスターにタグを追加することもできます。ElastiCache マネジメントコンソールを使用してクラスターを作成するときは、タグを追加できません。クラスターを作成した後は、コンソールを使用してクラスターにタグを追加できます。

AWS CLI を使用したタグの変更

AWS CLI を使用して、ElastiCache for Redis クラスターのノード上のタグを変更できます。

タグを変更するには:

- [add-tags-to-resource](#) を使用して、新しいタグを追加するか、既存のタグに関連付けられている値を変更します。
- [remove-tags-from-resource](#) を使用して、リソースから指定したタグを削除します。

どちらのオペレーションでも、指定のクラスターのタグとその値を示すリストが出力されます。

AWS CLI を使用したタグの削除

[remove-tags-from-resource](#) オペレーションを行い、AWS CLI を使用して ElastiCache for Redis クラスター内の既存のノードからタグを削除できます。

次のコードでは、AWS CLI を使用して、us-west-2 リージョンのクラスター my-cluster のノード my-cluster-001 から Service および Region というキーでタグを削除します。

Linux、macOS、Unix の場合:

```
aws elasticache remove-tags-from-resource \  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \  
  --tag-keys PM Service
```

Windows の場合:

```
aws elasticache remove-tags-from-resource ^  
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
```

```
--tag-keys PM Service
```

このオペレーションの出力は、次のオペレーションのリソースのすべてのタグを示した以下のリストのようになります。

```
{  
  "TagList": []  
}
```

詳細については、「AWS CLI for ElastiCache [remove-tags-from-resource](#)」を参照してください。

ElastiCache API を使用したコスト配分タグの管理

ElastiCache API を使用して、コスト配分タグを追加、変更、または削除できます。

コスト配分タグは、ElastiCache for Memcached クラスターに適用されます。タグ付けされるクラスターは、ARN (Amazon リソースネーム) を使用して指定されます。

サンプル `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

トピック

- [ElastiCache API を使用したタグの一覧表示](#)
- [ElastiCache API を使用したタグの追加](#)
- [ElastiCache API を使用したタグの変更](#)
- [ElastiCache API を使用したタグの削除](#)

ElastiCache API を使用したタグの一覧表示

[ListTagsForResource](#) オペレーションを行い、ElastiCache API を使用して既存のリソースのタグを一覧表示できます。

次のコードは、ElastiCache API を使用して、us-west-2 リージョンのリソース `my-cluster-001` のタグを一覧表示します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ListTagsForResource  
&ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001  
&SignatureVersion=4
```

```
&SignatureMethod=HmacSHA256
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

ElastiCache API を使用したタグの追加

[AddTagsToResource](#) オペレーションを行い、ElastiCache API を使用して、既存の ElastiCache クラスターにタグを追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されません。

次のコードは、ElastiCache API を使用して、us-west-2 リージョンのリソース my-cluster-001 に、Service と Region というキーを追加し、それぞれの値を elasticache と us-west-2 に設定します。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=AddTagsToResource
&ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Tags.member.1.Key=Service
&Tags.member.1.Value=elasticache
&Tags.member.2.Key=Region
&Tags.member.2.Value=us-west-2
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

詳細については、Amazon ElastiCache API リファレンスの「[AddTagsToResource](#)」を参照してください。

ElastiCache API を使用したタグの変更

ElastiCache API を使用して、ElastiCache クラスターのタグを変更できます。

タグの値を変更するには:

- [AddTagsToResource](#) オペレーションを使用して新しいタグと値を追加するか、既存のタグの値を変更します。
- [RemoveTagsFromResource](#) を使用して、リソースからタグを削除します。

どちらのオペレーションでも、指定のリソースのタグとその値を示すリストが出力されます。

[RemoveTagsFromResource](#) を使用して、リソースからタグを削除します。

ElastiCache API を使用したタグの削除

[RemoveTagsFromResource](#) オペレーションを行い、ElastiCache API を使用し、既存の ElastiCache for Redis ノードからタグを削除できます。

次のコードは、ElastiCache API を使用して、us-west-2 リージョンのクラスター my-cluster のノード my-cluster-001 から Service および Region というキーでタグを削除します。

```
https://elasticache.us-west-2.amazonaws.com/
?Action=RemoveTagsFromResource
&ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&TagKeys.member.1=Service
&TagKeys.member.2=Region
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

Amazon ElastiCache Well-Architected レンズの使用

このセクションでは、優れたアーキテクチャの ElastiCache ワークロードを設計するための設計原則とガイダンスの集合である Amazon ElastiCache Well-Architected レンズについて説明します。

- [ElastiCache レンズはAWS Well-Architected フレームワーク](#)に追加されたものです。
- 各柱には、ElastiCache アーキテクチャに関する議論を始めるのに役立つ一連の質問があります。
 - 各質問には、主なプラクティスとそのレポートスコアが記載されています。
 - 必須 - 本番前に必要 (リスクが高い場合を除く)
 - 最良 - カスタマーにとって最良の状態
 - 良い - カスタマーに推奨するもの (リスクが中程度ではない場合)
- Well-Architected の用語
 - [コンポーネント](#) — 組み合わせて要件を満たすコード、構成、AWS リソース。コンポーネントは他のコンポーネントと相互作用し、多くの場合、マイクロサービスアーキテクチャのサービスと同一視されます。

- [ワークロード](#) 一体となって事業価値をもたらす一連のコンポーネント。ワークロードの例としては、マーケティングウェブサイト、e コマースウェブサイト、モバイルアプリのバックエンド、分析プラットフォームなどがあります。

トピック

- [Amazon ElastiCache Well-Architected レンズのオペレーショナルエクセレンスの柱](#)
- [Amazon ElastiCache Well-Architected レンズのセキュリティの柱](#)
- [Amazon ElastiCache Well-Architected レンズの信頼性の柱](#)
- [Amazon ElastiCache Well-Architected レンズのパフォーマンス効率の柱](#)
- [Amazon ElastiCache Well-Architected レンズのコスト最適化の柱](#)

Amazon ElastiCache Well-Architected レンズのオペレーショナルエクセレンスの柱

運用上の優秀性の柱では、ビジネス価値をもたらし、プロセスと手順の継続的な向上を実現するために、システムを実行およびモニタリングすることに焦点を当てています。主なトピックは、変更の自動化、イベントへの対応、日常業務を管理するための標準の定義です。

トピック

- [OE 1: ElastiCache クラスターによってトリガーされるアラートやイベントをどのように把握して対応するか。](#)
- [OE 2: 既存の ElastiCache クラスターをいつ、どのようにスケーリングするか。](#)
- [OE 3: ElastiCache クラスターリソースを管理し、クラスターを最新の状態に保つ方法とは。](#)
- [OE 4: ElastiCache クラスターへのクライアントの接続をどのように管理するか。](#)
- [OE 5: ワークロード用に ElastiCache コンポーネントをどのようにデプロイするか。](#)
- [OE 6: 障害に対してどのように計画し、それを軽減するか。](#)
- [OE 7: Redis エンジンイベントをどのようにトラブルシューティングするか。](#)

OE 1: ElastiCache クラスターによってトリガーされるアラートやイベントをどのように把握して対応するか。

質問レベルの紹介: ElastiCache クラスターを運用している場合、特定のイベントが発生したときにオプションで通知やアラートを受け取ることができます。ElastiCache はデフォルトで、フェイル

オーバー、ノード交換、スケーリングオペレーション、定期メンテナンスなど、リソースに関連する [イベント](#) をログに記録します。各イベントには、日付と時刻、ソース名とソースタイプ、および説明が含まれます。

質問レベルのメリット: クラスターによって生成されたアラートをトリガーするイベントの背後にある根本的な理由を把握し、管理できると、より効果的に運用し、イベントに適切に対応できるようになります。

- [必須] ElastiCache コンソールでの ElastiCache により (リージョンを選択した後)、または [Amazon コマンドラインインターフェイス](#) (AWS CLI) の [describe-events](#) コマンドと [ElastiCache API](#) を使用して生成されたイベントを確認します。Amazon Simple Notification Service (Amazon SNS) を使用して重要なクラスターイベントの通知が送信されるように ElastiCache を設定します。クラスターで Amazon SNS を使用すると、ElastiCache イベントに対してプログラマ的にアクションを実行できます。
- イベントには、現在のイベントと予定されているイベントの 2 つの大きなカテゴリがあります。現在のイベントのリストには、リソースの作成と削除、スケーリングオペレーション、フェイルオーバー、ノードの再起動、スナップショットの作成、クラスターのパラメーターの変更、CA 証明書の更新、障害イベント (クラスタープロビジョニングの失敗 - VPC または ENI -、スケーリングの失敗 - ENI -、およびスナップショット障害) が含まれます。予定されているイベントのリストには、メンテナンス期間中に交換が予定されているノードとスケジュールが変更されたノード交換が含まれます。
- これらのイベントの中にはすぐに対応する必要がないものもありますが、最初にすべての障害イベントを確認することが重要です。
 - ElastiCache:AddCacheNodeFailed
 - ElastiCache:CacheClusterProvisioningFailed
 - ElastiCache:CacheClusterScalingFailed
 - ElastiCache:CacheNodesRebooted
 - ElastiCache:SnapshotFailed (Redis のみ)
- [リソース]:
 - [ElastiCache Amazon SNS 通知の管理](#)
 - [イベント通知と Amazon SNS](#)
- [最良] イベントへのレスポンスを自動化するには、SNS や Lambda 関数 などの AWS 製品やサービスの機能を活用します。ベストプラクティスに従って、小規模で頻繁に、元に戻せる変更をコードとして作成し、時間の経過に伴ってオペレーションを進化させます。クラスターをモニタリングするには Amazon CloudWatch メトリクスを使用する必要があります。

[リソース]: Lambda と SNS を使用するユースケースについては、[AWS Lambda](#)、[Amazon Route 53](#)、[Amazon SNS](#) を使用して [Amazon ElastiCache for Redis \(クラスターモードが無効\) リードレプリカエンドポイントをモニタリングします](#)。

OE 2: 既存の ElastiCache クラスターをいつ、どのようにスケールするか。

質問レベルの紹介: ElastiCache クラスターの適切なサイズ設定はバランスを図る作業であるため、基盤となるワークロードタイプに変更があるたびに評価する必要があります。目標は、ワークロードに適した規模の環境で運用することです。

質問レベルのメリット: リソースの使用率が高すぎると、レイテンシーが上昇し、全体的なパフォーマンスが低下する可能性があります。一方、十分に活用されていない場合、リソースのオーバプロビジョニングとなり、最適なコストで運用されない可能性があります。環境のサイズを適切に設定することで、パフォーマンス効率とコスト最適化のバランスを取ることができます。ElastiCache では、リソースの使用率が高すぎる、または低すぎる場合、2つの次元でスケールインすることで改善できます。ノード容量を増減することで垂直方向にスケールできます。ノードを追加および削除して、水平方向にスケールすることもできます。

- [必須] プライマリノードの CPU とネットワークの使用率が高すぎる場合は、読み取りオペレーションをレプリカノードにオフロードおよびリダイレクトすることで対処する必要があります。読み取り操作にはレプリカノードを使用して、プライマリノードの使用率を下げます。これは、Redis クライアントライブラリで構成でき、クラスターモードが無効な場合は ElastiCache リーダーエンドポイントに接続し、クラスターモードが有効な場合は Redis READONLY コマンドを使用します。

[リソース]:

- [接続エンドポイントの検索](#)
- [クラスターの適切なサイズ設定](#)
- [Redis の読み取り専用コマンド](#)
- [必須] CPU、メモリ、ネットワークなどの重要なクラスターリソースの使用状況をモニタリングします。これらの特定のクラスターリソースの使用率を追跡して、スケールアップの決定およびスケールアップオペレーションのタイプを通知する必要があります。ElastiCache for Redis クラスターモードが無効になっている場合、プライマリノードとレプリカノードは垂直方向にスケールできます。レプリカノードは、0 ノードから 5 ノードまで水平にスケールすることもできます。クラスターモードが有効になっている場合、同じことがクラスターの各シャードにも当てはまります。さらに、シャード数を増減できます。

[リソース]:

- [Amazon CloudWatch を使用して Amazon ElastiCache for Redis でベストプラクティスをモニタリングする](#)
- [ElastiCache for Redis クラスターのスケーリング](#)
- [ElastiCache for Memcached クラスターのスケーリング](#)
- [最良] 傾向を長期的にモニタリングすることで、特定の時点でモニタリングしても気付かないようなワークロードの変化を検出できます。長期的な傾向を検知するには、CloudWatch メトリクスを使用してより長時間の範囲をスキャンします。CloudWatch メトリクスを長期間観察して得た発見は、クラスターリソースの使用率に関する予測に役立つはずですが、CloudWatch のデータポイントとメトリクスは最大 455 日間利用できます。

[リソース]:

- [CloudWatch メトリクスによる ElastiCache for Redis のモニタリング](#)
- [CloudWatch メトリクスによる Memcached のモニタリング](#)
- [Amazon CloudWatch を使用して Amazon ElastiCache for Redis でベストプラクティスをモニタリングする](#)
- [最良] ElastiCache リソースが CloudFormation で作成されている場合は、運用上の一貫性を保ち、管理されていない構成変更やスタックドリフトを避けるために、CloudFormation テンプレートを使用して変更を実行するのがベストプラクティスです。

[リソース]:

- [CloudFormation の ElastiCache リソースタイプリファレンス](#)
- [最良] クラスターの運用データを使用してスケーリングオペレーションを自動化し、CloudWatch でしきい値を定義してアラームを設定します。CloudWatch イベントと Simple Notification Service (SNS) を使用して Lambda 関数をトリガーし、ElastiCache API を実行してクラスターを自動的にスケーリングします。例えば、EngineCPUUtilization メトリクスが長期間にわたって 80% に達したときにクラスターにシャードを追加します。また、メモリベースのしきい値として DatabaseMemoryUsedPercentages を使用することもできます。

[リソース]:

- [Amazon CloudWatch でのアラームの使用](#)
- [Amazon CloudWatch Events とは](#)
- [Amazon Simple Notification Service での AWS Lambda の使用](#)
- [ElastiCache API リファレンス](#)

OE 3: ElastiCache クラスターリソースを管理し、クラスターを最新の状態に保つ方法とは。

質問レベルの紹介:大規模に運用する場合、すべての ElastiCache リソースを細かく指摘して特定できることが不可欠です。新しいアプリケーション機能を展開する際には、開発、テスト、本番のすべての ElastiCache 環境タイプでクラスターバージョンの対称性を形成する必要があります。リソース属性を使用すると、新しい機能の展開や新しいセキュリティメカニズムの有効化など、運用上の目的に応じて環境を分けることができます。

質問レベルのメリット: 開発環境、テスト環境、本番環境を分離することが、運用上のベストプラクティスです。また、環境全体のクラスターとノードに、十分に理解され文書化されたプロセスを使用して最新のソフトウェアパッチを適用することもベストプラクティスです。ElastiCache のネイティブ機能を活用することで、エンジニアリングチームは ElastiCache のメンテナンスではなく、ビジネス目標の達成に集中できます。

- [最良] 入手可能な最新のエンジンバージョンで実行し、セルフサービスの更新が利用可能になったらすぐに適用します。ElastiCache は、指定したクラスターのメンテナンス期間中に、基盤となるインフラストラクチャを自動的に更新します。ただし、クラスターで実行されているノードは、セルフサービスの更新によって更新されます。これらの更新には、セキュリティパッチとマイナーソフトウェアの更新の 2 種類があります。パッチの種類の違いと適用時期について必ず理解しておいてください。

[リソース]:

- [Amazon ElastiCache でのセルフサービスの更新](#)
- [Amazon ElastiCache マネージドメンテナンスとサービスの更新のヘルプページ](#)
- [最良] タグを使用して ElastiCache リソースを整理します。タグは個々のノードではなくレプリケーショングループに使用します。リソースをクエリするときに表示するタグを設定したり、タグを使用して検索を実行したり、フィルターを適用できます。共通のタグセットを共有するリソースのコレクションを簡単に作成および管理するには、リソースグループを使用する必要があります。

[リソース]:

- [タグ付けのベストプラクティス](#)
- [CloudFormation の ElastiCache リソースタイプリファレンス](#)
- [パラメータグループ](#)

OE 4: ElastiCache クラスターへのクライアントの接続をどのように管理するか。

質問レベルの紹介: 大規模な運用を行う場合は、クライアントが ElastiCache クラスターに接続して、アプリケーションの運用面 (応答時間など) を管理する方法について理解する必要があります。

質問レベルのメリット: 最適な接続メカニズムを選択することで、タイムアウトなどの接続エラーによってアプリケーションが切断されることがなくなります。

- [必須] 読み取りオペレーションを書き込みオペレーションから分離し、レプリカノードに接続して読み取りオペレーションを実行します。ただし、書き込みを読み取りから分離すると、Redis レプリケーションの非同期性により、書き込み後すぐにキーを読み取ることができなくなることに注意してください。WAIT コマンドを利用すると、実際のデータの安全性が向上し、クライアントに応答する前にレプリカに書き込みを確認させることができますが、全体的なパフォーマンスは低下します。読み取りオペレーションにレプリカノードを使用することは、クラスターモードを無効にした ElastiCache リーダーエンドポイントを使用して、ElastiCache for Redis クライアントライブラリで設定できます。クラスターモードを有効にするには、ElastiCache for Redis READONLY コマンドを使用してください。ElastiCache for Redis クライアントライブラリの多くでは、ElastiCache for Redis READONLY はデフォルトで、または設定によって実装されています。

[リソース]:

- [接続エンドポイントの検索](#)
- [READONLY](#)
- [必須] 接続プーリングを使用します。TCP 接続を確立すると、クライアント側とサーバー側の両方で CPU 時間にコストがかかりますが、プーリングすると TCP 接続を再利用できます。

接続オーバーヘッドを減らすには、接続プーリングを使用する必要があります。接続のプールがあれば、アプリケーションは接続を「自由に」再利用および解放でき、接続を確立するコストを回避できます。接続プーリングは、アプリケーション環境で使用可能なフレームワークを使用して (サポートされている場合) ElastiCache for Redis クライアントライブラリを介して実装することも、ゼロから構築することもできます。

- [最良] クライアントのソケットタイムアウトが少なくとも 1 秒に設定されていることを確認します (一部のクライアントでは通常の「なし」のデフォルト設定)。
 - タイムアウト値の設定が低すぎると、サーバー負荷が高いときにタイムアウトする可能性があります。設定が高すぎると、アプリケーションが接続の問題を検出するのに長時間かかる可能性があります。

- クライアントアプリケーションに接続プーリングを実装して、新しい接続の量を制御します。これにより、接続の開閉、およびクラスターで TLS が有効になっている場合に TLS ハンドシェイクの実行に必要なレイテンシーと CPU 使用率が減少します。

[リソース]: [可用性を高めるために Amazon ElastiCache for Redis を設定します](#)

- [良い] パイプラインを使用すると (ユースケースで可能な場合)、パフォーマンスを大幅に向上させることができます。
- パイプラインを使用すると、アプリケーションクライアントとクラスター間の往復時間 (RTT) が短縮され、クライアントが以前のレスポンスをまだ読み取っていない場合でも、新しいリクエストを処理できます。
- パイプラインを使用すると、応答/ack を待たずに複数のコマンドをサーバーに送信できます。パイプラインの欠点は、最終的にすべてのレスポンスを一括取得したときに、エラーが発生しても、そのエラーを最後までキャッチできない可能性があることです。
- 不正なリクエストを省略したエラーが返されたときに、リクエストを再試行するメソッドを実装します。

[リソース]: [パイプライン](#)

OE 5: ワークロード用に ElastiCache コンポーネントをどのようにデプロイするか。

質問レベルの紹介: ElastiCache 環境は、AWS コンソールから手動でデプロイすることも、API、CLI、ツールキットなどを使用してプログラムでデプロイすることもできます。オペレーショナルエクセレンスのベストプラクティスでは、可能な限りコードを使用してデプロイメントを自動化することを推奨しています。さらに、ElastiCache クラスターはワークロードごとに分離することも、コストの最適化のために組み合わせることもできます。

質問レベルのメリット: ElastiCache 環境に最も適したデプロイメカニズムを選択することで、時間の経過とともにオペレーションエクセレンスを向上させることができます。ヒューマンエラーを最小限に抑え、再現性、柔軟性、イベントへの応答時間を向上させるため、可能な限りコードとしてオペレーションを実行することをお勧めします。

ワークロードの分離要件を理解することで、ワークロードごとに専用の ElastiCache 環境を使用するか、複数のワークロードを 1 つのクラスターにまとめるか、またはそれらを組み合わせるかを選択できます。トレードオフを理解することは、オペレーショナルエクセレンスとコスト最適化のバランスをとるのに役立ちます。

- [必須] ElastiCache で利用できるデプロイオプションを理解し、可能な限りこれらの手順を自動化します。自動化の手段として考えられるのは、CloudFormation、AWS CLI/SDK、API などです。

[リソース]:

- [Amazon ElastiCache リソースタイプのリファレンス](#)
- [elasticache](#)
- [Amazon ElastiCache API リファレンス](#)
- [必須] すべてのワークロードについて、必要なクラスター分離のレベルを決定します。
 - [最良]: 高度な分離 — ワークロードとクラスターの 1:1 のマッピング。ElastiCache リソースのアクセス、サイズ設定、スケーリング、管理をワークロードごとにきめ細かく制御できます。
 - [さらに良い]: 中程度の分離 — 目的別に分離されている、複数のワークロード (例えば、キャッシュワークロード専用のクラスターとメッセージング専用のクラスター) で共有されている可能性がある M: 1。
 - [良い]: 低度な分離 — 汎用タイプ、完全共有型の M:1。共有アクセスが許容されるワークロードに推奨されます。

OE 6: 障害に対してどのように計画し、それを軽減するか。

質問レベルの紹介: オペレーショナルエクセレンスには障害の予測が含まれ、定期的に「事前」演習を実施して潜在的な障害の原因を特定し、障害の排除や軽減を図ります。ElastiCache には、テスト目的でノード障害イベントをシミュレートできるフェイルオーバー API が用意されています。

質問レベルのメリット: 障害シナリオを事前にテストすることで、それらがワークロードにどのように影響するかを知ることができます。これにより、対応手順とその有効性を安全にテストできるだけでなく、チームはその実行に慣れておくことができます。

[必須] 開発/テストアカウントで定期的にフェイルオーバーテストを実行します。 [TestFailover](#)

OE 7: Redis エンジンイベントをどのようにトラブルシューティングするか。

質問レベルの紹介: オペレーショナルエクセレンスでは、サービスレベルとエンジンレベルの両方の情報を調査して、クラスターの状態とステータスを分析する能力が必要です。Amazon ElastiCache for Redis は、Amazon CloudWatch と Amazon Kinesis Data Firehose の両方に Redis エンジンログを送信できます。

質問レベルのメリット: Amazon ElastiCache for Redis クラスターで Redis エンジンログを有効にすると、クラスターの状態とパフォーマンスに影響するイベントに関する洞察が得られます。Redis エンジンログは、ElastiCache イベントメカニズムでは利用できないデータを Redis エンジンから直接

提供します。ElastiCache イベント (前述の OE-1 を参照) と Redis エンジンログの両方を注意深く観察することで、ElastiCache サービスの観点と Redis エンジンの観点の両方からトラブルシューティングを行うときにイベントの順序を決定できます。

- [必須] Redis エンジンのロギング機能が有効になっていることを確認します。この機能は ElastiCache for Redis 6.2 以降で使用可能です。これは、クラスターの作成中に実行することも、作成後にクラスターを変更することによって実行することもできます。
- Amazon CloudWatch Logs と Amazon Kinesis Data Firehose のどちらが Redis エンジンログの適切なターゲットであるかを判断します。
- CloudWatch または Kinesis Data Firehose 内の適切なターゲットログを選択して、ログを永続化します。クラスターが複数ある場合は、クラスターごとに異なるターゲットログを使用することを検討します。これにより、トラブルシューティング時にデータを分離しやすくなります。

[リソース]:

- ログ配信: [ログ配信](#)
- ロギング先: [Amazon CloudWatch Logs](#)
- Amazon CloudWatch Logs の紹介: [Amazon CloudWatch Logs とは](#)
- Amazon Kinesis Data Firehose の紹介: [Amazon Kinesis Data Firehose とは](#)
- [最良] Amazon CloudWatch Logs を使用する場合は、Amazon CloudWatch Logs Insights を活用して Redis エンジンログに重要な情報をクエリすることを検討します。

例えば、次のような LogLevel が「WARNING」のイベントを返す Redis エンジンログを含む CloudWatch ロググループに対してクエリを作成します。

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

[リソース]: [CloudWatch Logs Insights を使用したログデータの分析](#)

Amazon ElastiCache Well-Architected レンズのセキュリティの柱

セキュリティの柱は、情報とシステムの保護に焦点を当てています。主なトピックは、データの機密性と完全性、権限ベースの管理による誰が何を実行できるのかの特定と管理、システムの保護、セキュリティイベントを検出するための制御の確立です。

トピック

- [SEC 1: ElastiCache データへの許可されたアクセスを制御するためにどのような措置を講じているか。](#)
- [SEC 2: アプリケーションでは、ネットワークベースの制御に加えて、ElastiCache への追加の認証が必要か。](#)
- [SEC 3: コマンドが誤って実行され、データが失われたり失敗するリスクはあるか。](#)
- [SEC 4: ElastiCache を使用して保存中のデータの暗号化をどのように実現しているか。](#)
- [SEC 5: 転送中のデータを ElastiCache でどのように暗号化するか。](#)
- [SEC 6: コントロールプレーンリソースへのアクセスをどのように制限するか。](#)
- [SEC 7: セキュリティイベントをどのように検出して対応しているか。](#)

SEC 1: ElastiCache データへの許可されたアクセスを制御するためにどのような措置を講じているか。

質問レベルの紹介: すべての ElastiCache クラスターは、VPC 内の Amazon Elastic Compute Cloud インスタンス、サーバーレス関数 (AWS Lambda)、またはコンテナ (Amazon Elastic Container Service) からアクセスできるように設計されています。最もよく遭遇するシナリオは、同じ Amazon Virtual Private Cloud (Amazon Virtual Private Cloud) の Amazon Elastic Compute Cloud インスタンスから ElastiCache クラスターにアクセスすることです。Amazon EC2 インスタンスからクラスターに接続するには、Amazon EC2 インスタンスにクラスターへのアクセスを許可する必要があります。VPC で実行されている ElastiCache クラスターにアクセスするには、クラスターへのネットワーク進入を許可する必要があります。

質問レベルのメリット: クラスターへのネットワーク進入は VPC セキュリティグループによって制御されます。セキュリティグループは、Amazon EC2 インスタンスの仮想ファイアウォールとして機能し、受信トラフィックと送信トラフィックを制御します。インバウンドルールはインスタンスへの受信トラフィックを制御し、アウトバウンドルールはインスタンスからの送信トラフィックをコントロールします。ElastiCache の場合、クラスターを起動するときに、セキュリティグループを関連付ける必要があります。これにより、インバウンドとアウトバウンドのトラフィックルールがクラスターを構成するすべてのノードに適用されるようになります。さらに、ElastiCache はプライベートサブネットにのみデプロイするように設定されているため、VPC のプライベートネットワーク経由でのみアクセスできます。

- [必須] 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は、クラスターへのネットワークの進入とアクセスを制御します。デフォルトでは、セキュリティグループにはインバウンドルールが定義されていないため、ElastiCache への進入パスはありません。これを有効にするには、セキュリティグループのインバウンドルールを設定し、送信元 IP アドレス/範囲、TCP タイプの

トラフィック、および ElastiCache クラスターのポート (例えば、ElastiCache for Redis の場合はデフォルトポート 6379) を指定します。VPC 内のすべてのリソース (0.0.0.0/0) など、非常に広範囲の進入ソースを許可することは可能ですが、特定のセキュリティグループに関連付けられた Amazon Amazon EC2 インスタンスで実行されている Redis クライアントへのインバウンドアクセスのみを許可するなど、インバウンドルールはできるだけ細かく定義することをお勧めします。

[リソース]:

- [サブネットおよびサブネットグループ](#)
- [クラスターまたはレプリケーショングループへのアクセス](#)
- [セキュリティグループを使用してリソースへのトラフィックを制御する](#)
- [Linux インスタンス用の Amazon Elastic Compute Cloud セキュリティグループ](#)
- [必須] AWS Identity and Access Management ポリシーを AWS Lambda 関数に割り当てて、ElastiCache データへのアクセスを許可できます。この機能を有効にするには、AWSLambdaVPCAccessExecutionRole アクセス許可を持つ IAM 実行ロールを作成し、そのロールを AWS Lambda 関数に割り当てます。

[リソース]: Amazon VPC 内の Amazon ElastiCache にアクセスするための Lambda 関数の設定: [チュートリアル: Amazon VPC 内の Amazon ElastiCache にアクセスするための Lambda 関数の設定](#)

SEC 2: アプリケーションでは、ネットワークベースの制御に加えて、ElastiCache への追加の認証が必要か。

質問レベルの紹介: ElastiCache for Redis クラスターへのアクセスを個々のクライアントレベルで制限または制御する必要があるシナリオでは、ElastiCache for Redis AUTH コマンドを使用して認証することをお勧めします。ElastiCache for Redis 認証トークンとオプションのユーザーおよびユーザーグループ管理により、ElastiCache for Redis では、クライアントにコマンドやアクセスキーの実行を許可する前にパスワードを要求できるようになるため、データプレーンのセキュリティが向上します。

質問レベルのメリット: データを安全に保つために、ElastiCache for Redis にはデータへの不正アクセスを防ぐメカニズムが用意されています。これには、承認されたコマンドを実行する前にクライアントが ElastiCache に接続するために使用するロールベースのアクセス制御 (RBAC) AUTH または AUTH トークン (パスワード) を強制することが含まれます。

- [最良] ElastiCache for Redis 6.x 以降では、ユーザーグループ、ユーザー、アクセス文字列を定義して認証と承認の制御を定義します。ユーザーをユーザーグループに割り当ててから、ユーザーグ

ループをクラスターに割り当てます。RBAC を利用するには、クラスターの作成時に RBAC を選択し、転送中の暗号化を有効にする必要があります。RBAC を利用するには、TLS をサポートする Redis クライアントを使用していることを確認します。

[リソース]:

- [ElastiCache for Redis のレプリケーショングループへの RBAC の適用](#)
- [アクセス文字列を使用したアクセス許可の指定](#)
- [ACL](#)
- [サポートされている ElastiCache for Redis のバージョン](#)
- [最良] 6.x より前のバージョンの ElastiCache for Redis では、強力なトークンまたはパスワードを設定し、ElastiCache for Redis AUTH に厳格なパスワードポリシーを維持することに加えて、パスワードまたはトークンをローテーションすることがベストプラクティスです。ElastiCache は一度に最大 2 つの認証トークンを管理できます。また、クラスターを変更して、認証トークンの使用を明示的に要求することもできます。

[リソース]: [既存の ElastiCache for Redis クラスターでの AUTH トークンの変更](#)

SEC 3: コマンドが誤って実行され、データが失われたり失敗するリスクはあるか。

質問レベルの紹介: 誤って実行されたり、悪意のある攻撃者によって実行されたりすると、オペレーションに悪影響を及ぼす可能性のある Redis コマンドが多数あります。これらのコマンドは、パフォーマンスとデータ安全性の観点から、意図しない結果をもたらす可能性があります。例えば、開発者が開発環境で日常的に FLUSHALL コマンドを呼び出している場合、間違っても本番システムでこのコマンドを呼び出そうとすると、誤ってデータが失われる可能性があります。

質問レベルのメリット: ElastiCache 上の ElastiCache for Redis 5.0.3 から、ワークロードに支障をきたす可能性のある特定のコマンドの名前を変更できるようになりました。コマンドの名前を変更すると、クラスターでコマンドが誤って実行されるのを防ぐことができます。

- [必須]

[リソース]:

- [ElastiCache for Redis バージョン 5.0.3 \(廃止、バージョン 5.0.6 を使用してください\)](#)
- [Redis 5.0.3 パラメータの変更](#)
- [Redis セキュリティ](#)

SEC 4: ElastiCache を使用して保存中のデータの暗号化をどのように実現しているか。

質問レベルの紹介: ElastiCache for Redis はインメモリデータストアですが、クラスターの標準オペレーションの一部として (ストレージ上に) 永続化される可能性のあるすべてのデータを暗号化できます。これには、Amazon S3 に書き込まれたスケジュールバックアップと手動バックアップ、および同期およびスワップオペレーションの結果としてディスクストレージに保管されたデータが含まれます。M6g および R6g ファミリーのインスタンスタイプには、常時オンのインメモリ暗号化も備わっています。

質問レベルのメリット: ElastiCache for Redis では、データセキュリティを強化するため、保管時の暗号化をオプションで提供しています。

- [必須] 保管時の暗号化は、ElastiCache クラスター (レプリケーショングループ) に対してその作成時にのみ有効にできます。既存のクラスターを変更して、保管中のデータの暗号化を開始することはできません。デフォルトでは、保管中の暗号化に使用されるキーは ElastiCache が提供および管理します。

[リソース]:

- [保管時の暗号化の条件](#)
- [保管時の暗号化を有効にする](#)
- [最良] メモリ内にあるデータを暗号化する Amazon EC2 インスタンスタイプ (M6g や R6g など) を活用します。可能な場合は、保管中の暗号化に独自のキーを管理することを検討してください。より厳格なデータセキュリティ環境では、AWS Key Management Service (KMS) を使用してカスタマーマスターキー (CMK) を自己管理できます。AWS Key Management Service と ElastiCache を統合することで、ElastiCache for Redis クラスターの保管中のデータの暗号化に使用されるキーを作成、所有、管理できます。

[リソース]:

- [AWS Key Management Service からの KMS のカスタマーマネージドキーの使用](#)
- [AWS Key Management Service](#)
- [AWS KMS のコンセプト](#)

SEC 5: 転送中のデータを ElastiCache でどのように暗号化するか。

質問レベルの導入: 一般要件として、転送中のデータ漏えいを防止することが必要です。これには、分散システムのコンポーネント内のデータだけでなく、アプリケーションクライアントとクラス

ターノード間のデータも含まれます。ElastiCache for Redis では、クライアントとクラスター間、およびクラスターノード間の転送中のデータを暗号化できるようにすることで、この要件に対応します。M6g および R6g ファミリーのインスタンスタイプには、常時オンのインメモリ暗号化も備わっています。

質問レベルのメリット: Amazon ElastiCache 転送時の暗号化は、ある場所から別の場所に移動するときに、データの最も脆弱なポイントでのデータのセキュリティを強化できるオプション機能です。

- [必須] 転送中の暗号化は、作成時に ElastiCache for Redis クラスター (レプリケーショングループ) でのみ有効にできます。データの暗号化または復号化には追加の処理が必要なため、転送中の暗号化を実装すると、パフォーマンスにいくらか影響があることに注意してください。この影響について理解するため、転送中の暗号化を有効にする前と後にワークロードのベンチマークを行うことをお勧めします。

[リソース]:

- [転送時の暗号化の概要](#)

SEC 6: コントロールプレーンリソースへのアクセスをどのように制限するか。

質問レベルの導入: IAM ポリシーと ARN により、ElastiCache for Redis のよりきめ細かなアクセス制御が可能になり、ElastiCache for Redis クラスターの作成、変更、削除をより厳密に管理できるようになります。

質問レベルのメリット: レプリケーショングループ、ノードなどの Amazon ElastiCache リソースの管理を、IAM ポリシーに基づいて特定の権限を持つ AWS アカウントに制限できるため、リソースのセキュリティと信頼性が向上します。

- [必須] 特定の AWS Identity and Access Management ポリシーを AWS ユーザーに割り当てて Amazon ElastiCache リソースへのアクセスを管理し、どのアカウントがクラスターでどのアクションを実行できるかをより細かく制御できるようにします。

[リソース]:

- [ElastiCache リソースに対するアクセス許可の管理の概要](#)
- [Amazon ElastiCache でのアイデンティティベースのポリシー \(IAM ポリシー\) の使用](#)

SEC 7: セキュリティイベントをどのように検出して対応しているか。

質問レベルの紹介: ElastiCache は RBAC を有効にしてデプロイするとき、CloudWatch メトリクスをエクスポートしてユーザーにセキュリティイベントを通知します。これらのメトリクスは、接続する RBAC ユーザーに許可されていない認証、キーのアクセス、コマンドの実行の試みが失敗したことを特定するのに役立ちます。

さらに、AWS 製品とサービスのリソースは、デプロイを自動化し、すべてのアクションと変更を記録して後から確認または監査できるようにすることで、全体的なワークロードを保護するのに役立ちます。

質問レベルのメリット: イベントをモニタリングすることで、組織は要件、ポリシー、手順に従って対応できるようになります。これらのセキュリティイベントのモニタリングと対応を自動化すると、全体的なセキュリティ体制が強化されます。

- [必須] パブリッシュされている RBAC 認証と認証の失敗に関連する CloudWatch メトリクスをよく理解しておきます。
 - AuthenticationFailures = Redis への認証の試みに失敗
 - KeyAuthorizationFailures = ユーザーによる許可のないキーへのアクセスの試みの失敗
 - CommandAuthorizationFailures = ユーザーによる許可のないコマンドの実行の試みの失敗

[リソース]:

- [Redis のメトリクス](#)
- [最良] これらのメトリクスにアラートと通知を設定し、必要に応じて対応することをお勧めします。

[リソース]:

- [Amazon CloudWatch でのアラームの使用](#)
- [最良] Redis ACL LOG コマンドを使用して詳細を収集します

[リソース]:

- [ACL ログ](#)
- [最良] ElastiCache のデプロイとイベントのモニタリング、ロギング、分析に関連する AWS 製品とサービスの機能についてよく理解しておきます。

[リソース]:

- [AWS CloudTrail による Amazon ElastiCache API コールのロギング](#)

- [elasticache-redis-cluster-automatic-backup-check](#)
- [CloudWatch メトリクスの使用状況のモニタリング](#)

Amazon ElastiCache Well-Architected レンズの信頼性の柱

トピック

- [REL 1: 高可用性 \(HA\) アーキテクチャの導入をどのようにサポートしているか。](#)
- [REL 2: ElastiCache を使用して目標復旧時点 \(RPO\) をどのように達成しているか。](#)
- [REL 3: デイザスタリカバリ \(DR\) 要件をどのようにサポートしているか。](#)
- [REL 4: フェイルオーバーを効果的に計画するにはどうすればよいか。](#)
- [REL 5: ElastiCache コンポーネントがスケーリングに対応するように設計されがっているか。](#)

REL 1: 高可用性 (HA) アーキテクチャの導入をどのようにサポートしているか。

質問レベルの紹介: Amazon ElastiCache の高可用性アーキテクチャを理解することで、可用性イベントが発生しても回復力のある状態で運用できるようになります。

質問レベルのメリット: 障害に強い ElastiCache クラスターを構築することで、ElastiCache デプロイの可用性を高めることができます。

- [必須] ElastiCache クラスターに必要な信頼性のレベルを決定します。完全に一時的なワークロードからミッションクリティカルなワークロードまで、ワークロードが異なれば回復力の基準も異なります。開発、テスト、本番環境など、運用する環境のタイプごとにニーズを定義します。

キャッシュエンジン: Memcached と ElastiCache for Redis の比較

1. Memcached にはレプリケーションメカニズムがなく、主に一時的なワークロードに使用されません。
 2. ElastiCache for Redis は、以下で説明する HA 機能を提供します。
- [最良] HA を必要とするワークロードでは、1つのシャードしか必要としないスループット要件の低いワークロードでも、1シャードあたり少なくとも2つのレプリカがあるクラスターモードで ElastiCache for Redis を使用します。
 1. クラスターモードを有効にすると、マルチ AZ が自動的に有効になります。

マルチ AZ は、計画的または計画外のメンテナンスが発生した場合に、プライマリノードからレプリカへの自動フェイルオーバーを実行することでダウンタイムを最小限に抑え、AZ の障害を軽減します。

2. シャーディングされたワークロードの場合、最低 3 つのシャードを使用するとフェイルオーバーイベント中のリカバリが速くなります。これは、Redis Cluster Protocol では、クォーラムの達成にプライマリノードの過半数を利用可能にする必要があるためです。
3. アベイラビリティ全体で 2 つ以上のレプリカを設定します。

レプリカが 2 つあると、1 つのレプリカがメンテナンス中のとき、読み取りのスケーラビリティが向上し、シナリオでの読み取りの可用性も向上します。

4. Graviton2 ベースのノードタイプ (ほとんどのリージョンでのデフォルトノード) を使用します。

Amazon ElastiCache for Redis では、これらのノードでのパフォーマンスがさらに最適化されました。それにより、レプリケーションと同期のパフォーマンスが向上し、全体的な可用性が向上しています。

5. 予想されるトラフィックのピークに合わせてモニタリングし、適切なサイズに調整します。負荷が高いと、ElastiCache for Redis エンジンが応答しなくなり、可用性に影響する可能性があります。BytesUsedForCache および DatabaseMemoryUsagePercentage は、メモリ使用量を示す優れた指標で、ReplicationLag は、書き込み速度に基づくレプリケーションの状態を示す指標です。これらのメトリクスを使用してクラスタースケールングをトリガーできます。
6. [本番環境でフェイルオーバーイベントが発生する前にフェイルオーバー API](#) でテストすることで、クライアント側の耐障害性を確保します。

[リソース]:

- [可用性を向上するための Amazon ElastiCache for Redis の設定](#)
- [レプリケーショングループを使用する高可用性](#)

REL 2: ElastiCache を使用して目標復旧時点 (RPO) をどのように達成しているか。

質問レベルの紹介: ワークロードの RPO を理解して、ElastiCache のバックアップとリカバリの戦略に関する意思決定に役立っています。

質問レベルのメリット: RPO 戦略を策定することで、ディザスタリカバリシナリオが発生した場合の事業継続性を向上させることができます。バックアップと復元のポリシーを設計する

と、ElastiCache データの目標復旧時点 (RPO) の達成に役立ちます。ElastiCache for Redis には、Amazon S3 に保存されるスナップショット機能と、設定可能な保持ポリシーがあります。これらのスナップショットは、定義されたバックアップウィンドウ中に取得され、サービスによって自動的に処理されます。ワークロードにさらに細かいバックアップが必要な場合は、1 日あたり最大 20 の手動バックアップを作成できます。手動で作成されたバックアップにはサービス保持ポリシーがなく、無期限に保存できます。

- [必須] ElastiCache デプロイの RPO を理解し、文書化します。
 - Memcached はバックアッププロセスを提供していないことに注意してください。
 - ElastiCache のバックアップと復元の機能を確認してください。
- [最良] クラスターをバックアップするためのプロセスをしっかりと伝えておきます。
 - 必要に応じて手動バックアップを開始します。
 - 自動バックアップの保存ポリシーを確認します。
 - 手動バックアップは無期限に保持されることに注意してください。
 - 自動バックアップは使用率が低い時間帯にスケジュールします。
 - リードレプリカに対してバックアップオペレーションを実行して、クラスターのパフォーマンスへの影響を最小限に抑えます。
- [良い] ElastiCache のスケジュールバックアップ機能を活用して、定義した時間帯に定期的にデータをバックアップします。
 - バックアップからの復元を定期的にテストします。
- [リソース]:
 - [Redis](#)
 - [ElastiCache for Redis のバックアップと復元](#)
 - [手動バックアップの作成](#)
 - [自動バックアップのスケジュール](#)
 - [ElastiCache for Redis クラスターのバックアップと復元](#)

REL 3: デイザスタリカバリ (DR) 要件をどのようにサポートしているか。

質問レベルの紹介: デイザスタリカバリは、あらゆるワークロードプランニングの重要な側面です。ElastiCache for Redis には、ワークロード回復力要件に基づいてデイザスタリカバリを実装するためのオプションがいくつか用意されています。Amazon ElastiCache for Redis グローバルデータストアを使用すると、1 つのリージョンの ElastiCache for Redis クラスターに書き込み、そのデータを

他の 2 つのクロスリージョンレプリカクラスターから読み取ることができるため、リージョン間での低レイテンシーの読み取りとディザスタリカバリが可能になります。

質問レベルのメリット: さまざまな災害シナリオを理解して計画することで、事業継続性を確保できます。DR 戦略は、コスト、パフォーマンスへの影響、およびデータ損失の可能性とのバランスを取る必要があります。

- [必須] ワークロード要件に基づいて、すべての ElastiCache コンポーネントの DR 戦略を策定し、文書化します。ElastiCache がユニークなのは、完全に一時的で DR 戦略を必要としないユースケースもあれば、逆に、きわめて強固な DR 戦略を必要とするユースケースもあるという点です。すべてのオプションは、コストの最適化に対して比較検討する必要があります。回復力を高めるには、より多くのインフラストラクチャが必要です。

リージョンレベルおよびマルチリージョンレベルで利用可能な DR オプションを理解します。

- AZ 障害を防ぐために、マルチ AZ 配置が推奨されます。マルチ AZ アーキテクチャでは、必ずクラスターモードを有効にして、最低 3 つの AZ が利用可能な状態でデプロイします。
- リージョンの障害を防ぐために、グローバルデータストアの使用が推奨されます。
- [最良] リージョンレベルの耐障害性を必要とするワークロードには、グローバルデータストアを有効にします。
 - プライマリのパフォーマンスが低下した場合に備えて、セカンダリリージョンへのフェイルオーバーを計画します。
 - 本番環境でフェイルオーバーする前に、マルチリージョンのフェイルオーバープロセスをテストします。
 - ReplicationLag メトリクスをモニタリングして、フェイルオーバーイベント中のデータ損失の潜在的な影響を把握します。
- [リソース]:
 - [障害の軽減](#)
 - [グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)
 - [クラスターのサイズ変更 \(オプション\) によるバックアップからの復元](#)
 - [マルチ AZ による ElastiCache for Redis のダウンタイムの最小化](#)

REL 4: フェイルオーバーを効果的に計画するにはどうすればよいか。

質問レベルの紹介: 自動フェイルオーバーでマルチ AZ を有効にすることは、ElastiCache のベストプラクティスです。場合によっては、サービスオペレーションの一部として ElastiCache for Redis は

プライマリノードを置き換えます。例えば、計画されたメンテナンスのイベントや、ノードの障害またはアベイラビリティゾーンの問題という万が一の場合が含まれます。フェイルオーバーが成功するかどうかは、ElastiCache とクライアントライブラリの設定の両方に依存します。

質問レベルのメリット: ElastiCache フェイルオーバーのベストプラクティスを特定の ElastiCache for Redis クライアントライブラリと組み合わせて実行すると、フェイルオーバーイベント中の潜在的なダウンタイムを最小限に抑えることができます。

- [必須] クラスターモードが無効になっている場合は、タイムアウトを使用して、クライアントが古いプライマリノードから切断して、更新されたプライマリエンドポイントの IP アドレスを使用して新しいプライマリノードに再接続する必要があるかどうかを検出できるようにします。クラスターモードが有効になっている場合、クライアントライブラリは基盤となるクラスタートポロジの変更を検出します。これは、ほとんどの場合、ElastiCache for Redis クライアントライブラリの設定によって実現されます。これにより、更新の頻度と方法も設定できます。各クライアントライブラリには独自の設定があり、詳細は対応するドキュメントに記載されています。

[リソース]:

- [マルチ AZ による ElastiCache for Redis のダウンタイムの最小化](#)
- ElastiCache for Redis クライアントライブラリのベストプラクティスを確認してください。
- [必須] フェイルオーバーが成功するかどうかは、プライマリノードとレプリカノード間のレプリケーション環境が正常であるかどうかによります。Redis レプリケーションの非同期性と利用可能な CloudWatch メトリクスをレビューして理解し、プライマリノードとレプリカノード間のレプリケーションラグについてレポートします。データの安全性を高める必要があるユースケースでは、Redis WAIT コマンドを活用して、接続されたクライアントに応答する前にレプリカに書き込みの確認を強制します。

[リソース]:

- [Redis のメトリクス](#)
- [Amazon CloudWatch を使用して Amazon ElastiCache for Redis でベストプラクティスをモニタリングする](#)
- [最良] ElastiCache テストフェイルオーバー API を使用して、フェイルオーバー中のアプリケーションの応答性を定期的に検証します。

[リソース]:

- [Amazon ElastiCache for Redis でのリードレプリカへの自動フェイルオーバーのテスト](#)
- [自動フェイルオーバーのテスト](#)

REL 5: ElastiCache コンポーネントがスケーリングに対応するように設計されがっているか。

質問レベルの紹介: スケーリング機能と利用可能なデプロイトポロジを理解することで、ElastiCache コンポーネントは変化するワークロード要件に合わせて経時的に調整できます。ElastiCache には、イン/アウト (水平) とアップ/ダウン (垂直) の 4 通りのスケーリングがあります。

質問レベルのメリット: ElastiCache デプロイのベストプラクティスに従うと、スケーリングの柔軟性が最大化されるだけでなく、障害の影響を最小限に抑えるために水平方向にスケーリングするという Well Architected の原則も満たすことができます。

- [必須] クラスターモード有効ポロジとクラスターモード無効トポロジの違いを理解します。ほとんどの場合、クラスターモードを有効にしてデプロイすることが推奨されます。これは、時間の経過とともにスケーラビリティが向上できるためです。クラスターモードが無効になっているコンポーネントでは、リードレプリカを追加することにより水平方向にスケーリングする機能に制限があります。
- [必須] いつ、どのようにスケーリングすべきかを理解します。
 - READIOPS を増やすには、レプリカを追加します
 - WRITEOPS を増やすには、シャードを追加します (スケールアウト)
 - ネットワーク IO を増やすには — ネットワーク最適化インスタンス、スケールアップを使用します
- [最良] ElastiCache コンポーネントをクラスターモードを有効にしてデプロイし、少ないノード数で大規模にするのではなく、小さなノードを数多くします。これにより、ノード障害時の影響範囲を効果的に制限できます。
- [最良] クラスターにレプリカを含めると、スケーリングイベント中の応答性が向上します
- [良い] クラスターモードが無効になっている場合は、リードレプリカを活用して全体的な読み取り容量を増加します。ElastiCache は、クラスターモードが無効な状態で最大 5 つのリードレプリカをサポートし、垂直スケーリングもサポートします。
- [リソース]:
 - [ElastiCache for Redis クラスターのスケーリング](#)
 - [オンラインスケールアップ](#)
 - [ElastiCache for Memcached クラスターのスケーリング](#)

Amazon ElastiCache Well-Architected レンズのパフォーマンス効率の柱

パフォーマンス効率の柱は、IT リソースとコンピューティングリソースを効率的に使用することに重点を置いています。主なトピックには、ワークロード要件に基づいた適切なリソースの種類とサイズを選択、パフォーマンスの監視、ビジネスニーズの変化に応じて効率を維持するための情報に基づいた意思決定が含まれます。

トピック

- [PE 1: Amazon ElastiCache クラスターのパフォーマンスをどのようにモニタリングするか。](#)
- [PE 2: ElastiCache クラスターノード間で作業をどのように分散するか。](#)
- [PE 3: キャッシュワークロードの場合、キャッシュの有効性とパフォーマンスをどのように追跡して報告するか。](#)
- [PE 4: ワークロードがどのようにしてネットワークリソースと接続の使用を最適化するか。](#)
- [PE 5: キーの削除および/またはエビクションをどのように管理しているか。](#)
- [PE 6: ElastiCache のデータをどのようにモデル化して操作するか。](#)
- [PE 7: 実行速度が遅いコマンドをどのように Amazon ElastiCache クラスターに記録するか。](#)
- [PE8: 自動スケーリングは ElastiCache クラスターのパフォーマンスを向上させるのにどのように役立つか。](#)

PE 1: Amazon ElastiCache クラスターのパフォーマンスをどのようにモニタリングするか。

質問レベルの紹介: 既存のモニタリングメトリクスを理解することで、現在の使用率を特定できます。適切にモニタリングすることで、クラスターのパフォーマンスに影響を与える潜在的なボトルネックを特定できます。

質問レベルのメリット: クラスターに関連するメトリクスを理解することは、レイテンシーの削減とスループットの向上につながる最適化手法の指針となります。

- [必須] ワークロードのサブセットを使用したベースラインパフォーマンスのテスト。
 - 負荷テストなどのメカニズムを使用して、実際のワークロードのパフォーマンスをモニタリングする必要があります。
 - これらのテストを実行しながら CloudWatch メトリクスをモニタリングして、利用可能なメトリクスを理解し、パフォーマンスのベースラインを確立します。

- [最良] ElastiCache for Redis のワークロードでは、KEYS など、計算量の多いコマンドの名前を変更して、ユーザーが本番クラスターでブロッキングコマンドを実行する能力を制限します。
- エンジン 6.x を実行する ElastiCache for Redis ワークロードでは、ロールベースのアクセス制御を利用して特定のコマンドを制限できます。コマンドへのアクセスは、AWS コンソールまたは CLI を使用してユーザーとユーザーグループを作成し、ユーザーグループを ElastiCache for Redis クラスターに関連付けることで制御できます。Redis 6 では、RBAC が有効になっている場合、「-@dangerous」を使用できます。これにより、そのユーザーには KEYS、MONITOR、SORT などのコストの高いコマンドが許可されなくなります。
- エンジンバージョン 5.x では、Amazon ElastiCache for Redis クラスターパラメータグループで `rename-commands` パラメータを使用してコマンドの名前を変更します。
- [さらに良い] 処理が遅いクエリを分析し、最適化手法を検討します。
 - ElastiCache for Redis ワークロードの場合は、スローログを分析してクエリについて詳しく調べます。例えば、`redis-cli slowlog get 10` コマンドを使用して、遅延のしきい値 (デフォルトは 10 秒) を超えた直近の 10 個のコマンドを表示できます。
 - 特定のクエリは、複雑な ElastiCache for Redis データ構造を使用するとより効率的に実行できます。例えば、数値形式の範囲検索の場合、アプリケーションはソートセットを使用して単純な数値インデックスを実装できます。これらのインデックスを管理することで、データセットに対して実行されるスキャン数を減らし、より高いパフォーマンス効率でデータを返すことができます。
 - ElastiCache for Redis ワークロードでは、`redis-benchmark` は、クライアント数やデータサイズなどのユーザー定義の入力を使用し、さまざまなコマンドのパフォーマンスをテストするシンプルなインターフェイスを提供します。
 - Memcached はシンプルなキーレベルコマンドのみをサポートしているため、クライアントのクエリを処理するためにキースペースを繰り返し処理しないように、追加のキーをインデックスとして作成することを検討してください。
- [リソース]:
 - [CloudWatch メトリクスの使用状況のモニタリング](#)
 - [CloudWatch メトリクスの使用状況のモニタリング](#)
 - [Amazon CloudWatch でのアラームの使用](#)
 - [Redis 固有のパラメータ](#)
 - [SLOWLOG](#)
 - [Redis ベンチマーク](#)

PE 2: ElastiCache クラスターノード間で作業をどのように分散するか。

質問レベルの紹介: アプリケーションを Amazon ElastiCache ノードに接続する方法は、クラスターのパフォーマンスとスケーラビリティに影響を与える可能性があります。

質問レベルのメリット: クラスター内の利用可能なノードを適切に使用することで、利用可能なリソース全体に作業が分散されます。次の手法は、リソースのアイドル状態を回避するのにも役立ちます。

- [必須] クライアントを適切な ElastiCache エンドポイントに接続します。
 - Amazon ElastiCache for Redis は、使用中のクラスターモードに基づいてさまざまなエンドポイントを実装します。クラスターモードを有効にすると、ElastiCache が設定エンドポイントを提供します。クラスターモードを無効にすると、ElastiCache はプライマリエンドポイント (通常は書き込み用) と、レプリカ間の読み取りのバランスを図るリーダーエンドポイントを提供します。これらのエンドポイントを正しく実装すると、パフォーマンスが向上し、オペレーションのスケールアップが容易になります。特定の要件がない限り、個々のノードエンドポイントへの接続は回避します。
 - マルチノードの Memcached クラスターでは、ElastiCache は自動検出を可能にする設定エンドポイントを提供します。キャッシュノード全体に作業を均等に分散するには、ハッシュアルゴリズムの使用をお勧めします。多くの Memcached クライアントライブラリは整合性のあるハッシュを実装しています。使用中のライブラリのドキュメントを参照し、整合性のあるハッシュをサポートするかどうかと、その実装方法について確認してください。これらの機能の実装の詳細については、[こちら](#)をご覧ください。
- [さらに良い] スケーラビリティを向上させるために、ElastiCache for Redis クラスターモードを有効に活用します。
 - ElastiCache for Redis (クラスターモード有効) クラスターは [オンラインスケールアップ/ダウン](#) (アウト/イン、アップ/ダウン) をサポートしているため、シャード全体にデータを動的に分散できます。設定エンドポイントを使用すると、クラスター対応クライアントがクラスターポロジの変化に確実に適応できるようになります。
 - また、ElastiCache for Redis (クラスターモードが有効) クラスター内の使用可能なシャード間でハッシュスロットを移動して、クラスターのバランスを再調整することもできます。これにより、使用可能なシャード間で作業をより効率的に分散できます。
- [さらに良い] ワークロード内のホットキーを特定して修正するための戦略を実装します。
 - リスト、ストリーム、セットなどの多次元の Redis データ構造の影響を考慮してください。これらのデータ構造は、単一のノードにある単一の Redis Keys に保存されます。多次元キーが非常に大きい場合、他のデータ型よりも多くのネットワーク容量とメモリを使用する可能性があります。

り、そのノードが不均衡に使用される可能性があります。可能な場合は、データアクセスを多数の個別のキーに分散するようにワークロードを設計します。

- ワークロード内のホットキーは、使用中のノードのパフォーマンスに影響を与える可能性があります。ElastiCache for Redis のワークロードでは、LFU 最大メモリポリシーが設定されている場合、`redis-cli --hotkeys` を使用してホットキーを検出できます。
- ホットキーを複数のノードに複製して、アクセス権を均等に分散することを検討してください。この方法では、クライアントは複数のプライマリノードへの書き込み (Redis ノード自体はこの機能は提供しません)、元のキー名に加えて読み取り元のキー名のリストの管理が必要となります。
- ElastiCache for Redis バージョン 6 は、サーバー支援 [クライアント側キャッシュ](#) をサポートしています。これにより、アプリケーションはキーが変更されるのを待ってから、ElastiCache にネットワークコールバックを行うことができます。
- [リソース]:
 - [可用性を向上するための Amazon ElastiCache for Redis の設定](#)
 - [接続エンドポイントの検索](#)
 - [負荷分散のベストプラクティス](#)
 - [Redis 用のオンラインリシャーディングおよびシャードの再分散 \(クラスターモードが有効\)](#)
 - [Redis でのクライアント側キャッシュ](#)

PE 3: キャッシュワークロードの場合、キャッシュの有効性とパフォーマンスをどのように追跡して報告するか。

質問レベルの紹介: キャッシュは ElastiCache で一般的に発生するワークロードであり、キャッシュの有効性とパフォーマンスを管理する方法を理解することが重要です。

質問レベルのメリット: アプリケーションのパフォーマンスが低下する兆候が見られる場合があります。キャッシュワークロードでは、キャッシュ固有のメトリクスを使用してアプリケーションのパフォーマンスを向上させる方法を決定できることが重要です。

- [必須] キャッシュヒットレートを経時的に測定し、追跡します。キャッシュの効率は、その「キャッシュヒットレート」によって決まります。キャッシュヒットレートは、キーヒット数の合計をヒット数とミス数の合計で割った値で定義されます。比率が 1 に近いほど、キャッシュの効率は高くなります。キャッシュヒットレートが低いのは、キャッシュミスの数が多いためです。キャッシュミスは、要求されたキーがキャッシュに見つからない場合に発生します。キーは、エビクションまたは削除されたか、有効期限が切れているか、あるいは存在したことがないため、

キャッシュに存在しません。キーがキャッシュにない理由を理解し、キーをキャッシュに含めるための適切な戦略を立てます。

[リソース]:

- [Redis のメトリクス](#)
- [必須] アプリケーションキャッシュのパフォーマンスをレイテンシーや CPU 使用率の値と合わせて測定および収集し、存続時間やその他のアプリケーションコンポーネントを調整する必要があるかどうかを判断します。ElastiCache では、データ構造ごとにレイテンシーを集約した一連の CloudWatch メトリクスを提供しています。これらのレイテンシーメトリクスは ElastiCache for Redis INFO コマンドの commandstats 統計を使用して計算され、ネットワークと I/O 時間は含まれていません。これは、ElastiCache for Redis がオペレーションを処理するのにかかる時間のみです。

[リソース]:

- [Redis のメトリクス](#)
- [Amazon CloudWatch を使用して Amazon ElastiCache for Redis でベストプラクティスをモニタリングする](#)
- [最良] ニーズに合った適切なキャッシュ戦略を選択します。キャッシュヒットレートが低いのは、キャッシュミス数が多いためです。ワークロードがキャッシュミス数が少ないように設計されている場合 (リアルタイム通信など)、キャッシュ戦略を見直し、メモリやパフォーマンスを測定するためのクエリ計測など、ワークロードに最適な解決策を適用するのが最善です。キャッシュを入力し維持するために実装する実際の戦略は、クライアントがキャッシュする必要のあるデータとそのデータへのアクセスパターンによって決まります。例えば、ストリーミングアプリケーションのパーソナライズされた推奨とトレンドニュースの両方に同じ戦略を使用する可能性はほとんどありません。

[リソース]:

- [キャッシュ戦略](#)
- [キャッシュのベストプラクティス](#)
- [Amazon ElastiCache ホワイトペーパーによるスケールに応じたパフォーマンス](#)

PE 4: ワークロードがどのようにしてネットワークリソースと接続の使用を最適化するか。

質問レベルの紹介: ElastiCache for Redis と Memcached は多くのアプリケーションクライアントでサポートされており、実装は異なる場合があります。潜在的なパフォーマンスへの影響を分析するには、実施されているネットワークと接続の管理について理解する必要があります。

質問レベルのメリット: ネットワークリソースを効率的に使用することで、クラスターのパフォーマンス効率を向上させることができます。以下の推奨事項は、ネットワークの需要を減らし、クラスターのレイテンシーとスループットを向上させることができます。

- [必須] ElastiCache クラスターへの接続を積極的に管理します。
 - アプリケーション内の接続プーリングにより、接続の開閉により生じるクラスターのオーバーヘッドの量を減らすことができます。CurrConnections と NewConnections を使用して Amazon CloudWatch の接続動作をモニタリングします。
 - 必要に応じてクライアント接続を適切に閉じて接続リークを回避します。接続管理戦略には、使用されていない接続を適切に閉じることや、接続タイムアウトを設定することが含まれます。
 - Memcached ワークロードの場合、memcached_connections_overhead と呼ばれる接続を処理するために確保される設定可能なメモリの量があります。
- [さらに良い] 大きなオブジェクトを圧縮してメモリを減らし、ネットワークのスループットを向上させます。
 - データを圧縮すると、必要なネットワークスループット (Gbps) は低下しますが、データを圧縮および解凍するアプリケーションでの作業量が増えます。
 - 圧縮により、キーが消費するメモリの量も減少します。
 - アプリケーションのニーズに応じて、圧縮率と圧縮速度のトレードオフを検討してください。
- [リソース]:
 - [Amazon ElastiCache for Redis — グローバルデータストア](#)
 - [Memcached 固有のパラメータ](#)
 - [Amazon ElastiCache for Redis 5.0.3 は I/O 処理を強化してパフォーマンスを向上](#)
 - [Redis のメトリクス](#)
 - [可用性を向上するための Amazon ElastiCache for Redis の設定](#)

PE 5: キーの削除および/またはエビクションをどのように管理しているか。

質問レベルの紹介: ワークロードにはさまざまな要件があり、クラスターノードがメモリ消費量の上限に近づいたときに予想される動作も異なります。Amazon ElastiCache for Redis には、このような状況に対処するためのさまざまなポリシーがあります。

質問レベルのメリット: 使用可能なメモリを適切に管理し、エビクションポリシーを理解することで、インスタンスのメモリ制限を超えた場合のクラスターの動作を確実に把握できます。

- [必須] データアクセスを実装して、どのポリシーを適用するかを評価します。クラスターでエビクションを実行するかどうか、またその方法を制御するための適切な最大メモリーポリシーを特定します。
 - エビクションは、クラスターの最大メモリーが消費され、エビクションを許可するポリシーが設定されているときに発生します。この状況でのクラスターの動作は、指定されたエビクションポリシーによって異なります。このポリシーは、ElastiCache for Redis クラスターパラメータグループで `maxmemory-policy` を使用して管理できます。
 - デフォルトのポリシー `volatile-lru` は、有効期限 (TTL 値) が設定されたキーをエビクションすることでメモリを解放します。最も使用頻度が低い (LFU) ポリシーと最も最近使用されていない (LRU) ポリシーは、使用状況に基づいてキーを削除します。
 - Memcached ワークロードの場合、各ノードのエビクションを制御するデフォルトの LRU ポリシーが設定されています。Amazon ElastiCache クラスター上のエビクションの数は、Amazon CloudWatch のエビクションメトリクスを使用してモニタリングできます。
- [さらに良い] 削除動作を標準化してクラスターのパフォーマンスへの影響を制御し、予期しないパフォーマンスのボトルネックを回避します。
 - ElastiCache for Redis のワークロードでは、クラスターからキーを明示的に削除すると、`UNLINK` は `DEL` のように、指定されたキーが削除されます。ただし、このコマンドは実際のメモリ再利用を別のスレッドで実行するため、ブロッキングしませんが、`DEL` はします。実際の削除は後に非同期で行われます。
 - ElastiCache for Redis 6.x ワークロードでは、`DEL` コマンドの動作は、`lazyfree-lazy-user-de1` パラメータを使用してパラメータグループで変更できます。
- [リソース]:
 - [パラメータグループを使用したエンジンパラメータの設定](#)
 - [UNLINK](#)
 - [AWS によるクラウド財務管理](#)

PE 6: ElastiCache のデータをどのようにモデル化して操作するか。

質問レベルの紹介: ElastiCache は、使用するデータ構造とデータモデルにアプリケーションが大きく依存しますが、基盤となるデータストア (存在する場合) も考慮する必要があります。利用可能な ElastiCache for Redis のデータ構造を理解し、ニーズに最も適したデータ構造を使用していることを確認します。

質問レベルのメリット: ElastiCache のデータモデリングには、アプリケーションのユースケース、データタイプ、データ要素間の関係など、複数のレイヤーがあります。さらに、ElastiCache for Redis の各データタイプとコマンドには、適切に文書化された独自のパフォーマンスシグネチャがあります。

- [最良] ベストプラクティスは、意図しないデータの上書きを減らすことです。キー名の重複を最小限に抑える命名規則を使用します。従来のデータ構造の命名では、APPNAME:CONTEXT:ID、ORDER-APP:CUSTOMER:123 などの階層的な方法を使用します。

[リソース]:

- [キー命名](#)
- [最良] ElastiCache for Redis コマンドの時間の複雑さは Big O 表記法で定義されます。このコマンドの時間の複雑さは、その影響をアルゴリズム的または数学的に表現したものです。アプリケーションに新しいデータ型を導入する際には、関連するコマンドの時間の複雑さを注意深く見直す必要があります。時間の複雑さが $O(1)$ のコマンドは時間が一定で、入力のサイズには依存しませんが、時間の複雑さが $O(N)$ のコマンドは時間が線形で、入力のサイズに影響されます。ElastiCache for Redis はシングルスレッド設計であるため、時間が複雑なオペレーションを大量に行うと、パフォーマンスが低下し、オペレーションがタイムアウトする可能性があります。

[リソース]:

- [コマンド](#)
- [最良] API を使用すると、クラスター内のデータモデルを GUI で表示できます。

[リソース]:

- [Redis Commander](#)
- [Redis Browser](#)
- [Redsmin](#)

PE 7: 実行速度が遅いコマンドをどのように Amazon ElastiCache クラスターに記録するか。

質問レベルの紹介: 実行時間の長いコマンドのキャプチャ、集約、通知によるパフォーマンスチューニングのメリット。コマンドの実行に必要な時間を把握することで、パフォーマンスを低下させているコマンド、またエンジンの最適な実行を妨げるコマンドを特定できます。Amazon ElastiCache for Redis には、この情報を Amazon CloudWatch または Amazon Kinesis Data Firehose に転送する機能もあります。

質問レベルのメリット: 専用の場所にログを記録し、処理が遅いコマンドに通知イベントを提供することは、詳細なパフォーマンス分析に役立ち、自動イベントのトリガーにも使用できます。

- [必須] Amazon ElastiCache for Redis のエンジンバージョン 6.0 以降を実行し、パラメータグループが適切に設定され、クラスターで SLOWLOG ロギングが有効になっていること。
 - 必須パラメータは、エンジンバージョンの互換性が Redis バージョン 6.0 以上に設定されている場合にのみ使用できます。
 - SLOWLOG ロギングは、コマンドのサーバー実行時間が指定された値よりも長い場合に発生します。クラスターの動作は、関連するパラメータグループのパラメータ (`slowlog-log-slower-than` および `slowlog-max-len`) によって異なります。
 - 変更は即時適用されます。
- [最良] CloudWatch または Kinesis Data Firehose の機能を活用します。
 - CloudWatch、CloudWatch Logs Insights、Amazon Simple Notification Services のフィルタリング機能とアラーム機能を使用して、パフォーマンスのモニタリングとイベント通知を行います。
 - Kinesis Data Firehose のストリーミング機能を使用して、SLOWLOG ログを永続ストレージにアーカイブしたり、クラスターパラメータの自動チューニングをトリガーします。
 - JSON 形式とプレーンテキスト形式のどちらがニーズに最も適しているかを判断してください。
 - CloudWatch または Kinesis Data Firehose に公開するための IAM アクセス許可を提供します。
- [さらに良い] `slowlog-log-slower-than` をデフォルト以外の値に設定します。
 - このパラメータは、実行速度が遅いコマンドとして記録されるまでにコマンドが Redis エンジン内で実行できる時間を決定します。デフォルト値は 10,000 マイクロ秒 (10 ミリ秒) です。一部のワークロードでは、このデフォルト値は高すぎる場合があります。
 - アプリケーションのニーズとテスト結果に基づいて、ワークロードにより適した値を決定します。ただし、値が低すぎると、過剰なデータが生成される可能性があります。
- [さらに良い] `slowlog-max-len` をデフォルト値のままにしておきます。

- このパラメータは、所定の時間において実行速度が遅いコマンドを Redis メモリにキャプチャする数の上限を決定します。値が 0 の場合、キャプチャは事実上無効になります。値が大きいほど、メモリに保存されるエントリの数が増え、確認する前に重要な情報がエビクションされる可能性が低くなります。デフォルト値は 128 です。
- デフォルト値は、ほとんどのワークロードに適しています。redis-cli から SLOWLOG コマンドを使用して拡張された時間枠でデータを分析する必要がある場合は、この値の増加を検討してください。これにより、より多くのコマンドを Redis メモリに残すことができます。

SLOWLOG データを CloudWatch Logs または Kinesis Data Firehose に送信する場合、データは永続化され、ElastiCache システムの外部で分析できるため、実行速度が遅い多数のコマンドを Redis メモリに保存する必要がなくなります。

- [リソース]:
 - [ElastiCache for Redis キャッシュクラスターで Redis Slow ログを有効にするには](#)
 - [ログ配信](#)
 - [Redis 固有のパラメータ](#)
 - <https://aws.amazon.com/cloudwatch/> Amazon CloudWatch
 - [Amazon Kinesis Data Firehose](#)

PE8: 自動スケーリングは ElastiCache クラスターのパフォーマンスを向上させるのにどのように役立つか。

質問レベルの紹介: Redis 自動スケーリングの機能を実装することで、ElastiCache コンポーネントは時間の経過とともに調整され、必要なシャードやレプリカを自動的に増減できます。これは、ターゲットトラッキングポリシーまたはスケジュールされたスケーリングポリシーのいずれかを実装することで実現できます。

質問レベルのメリット: ワークロードの急増を把握し、計画を立てることで、キャッシュのパフォーマンスと事業継続性を確実に向上させることができます。ElastiCache for Redis 自動スケーリングは、クラスターが希望するパフォーマンスレベルで動作していることを確認するために、CPU とメモリの使用率を継続的にモニタリングします。

- [必須] ElastiCache for Redis に対してクラスターを起動する場合、
 1. クラスターモードが有効になっていることを確認します
 2. インスタンスが自動スケーリングをサポートする特定のタイプとサイズのファミリーに属していることを確認します

3. クラスターがグローバルデータストア、Outposts、または Local Zones で実行されていないことを確認します

[リソース]:

- [Redis \(クラスターモードが有効\) でのクラスターのスケールリング](#)
 - [シャードでの自動スケールリングの使用](#)
 - [レプリカでの自動スケールリングの使用](#)
- [最良] ワークロードが読み取り中心か、または書き込み中心かを特定して、スケールリングポリシーを定義します。最高のパフォーマンスを達成するには、1つのトラッキングメトリクスのみを使用します。自動スケールリングポリシーは、ターゲットがヒットするとスケールアウトしますが、すべてのターゲットトラッキングポリシーがスケールインできる状態になって初めてスケールインするため、ディメンションごとに複数のポリシーを設定するのは避けることをお勧めします。

[リソース]:

- [自動スケールリングポリシー](#)
 - [スケールリングポリシーの定義](#)
- [最良] パフォーマンスを経時的にモニタリングすることで、特定の時点でモニタリングしても気付かないようなワークロードの変化を検出できます。4週間のクラスター使用率の対応する CloudWatch メトリクスを分析し、ターゲットのしきい値を決定できます。選択する値が不明な場合は、サポートされる最小定義メトリクス値から開始することをお勧めします。

[リソース]:

- [CloudWatch メトリクスの使用状況のモニタリング](#)
- [より良い] スケールリングポリシーを策定し、可用性の問題を軽減するために、クラスターに必要なシャード/レプリカの正確な数を特定するために、予想される最小ワークロードと最大ワークロードでアプリケーションをテストすることをお勧めします。

[リソース]:

- [スケラブルなターゲットの登録](#)
- [スケラブルなターゲットの登録](#)

Amazon ElastiCache Well-Architected レンズのコスト最適化の柱

コスト最適化の柱は、不必要なコストを回避することに焦点を当てています。主なトピックには、資金の用途の把握と管理、最適なノードタイプの選択 (ワークロードのニーズに基づくデータ階層化を

サポートするインスタンスの使用)、適切な数のリソースタイプ (リードレプリカの数)、経時的な支出の分析、浪費することなくビジネスニーズを満たすためのスケーリングなどがあります。

トピック

- コスト 1: ElastiCache リソースに関連するコストをどのように特定し、追跡するか。作成したリソースをユーザーが作成、管理、廃棄できるようにするメカニズムをどのように開発しますか。
- コスト 2: ElastiCache リソースに関連するコストの最適化のために、継続的モニタリングツールをどのように使用するか。
- コスト 3: データ階層化をサポートするインスタンスタイプを使用すべきか。データ階層化のメリットは何か。データ階層化インスタンスを使用すべきでないのはどのような場合か。

コスト 1: ElastiCache リソースに関連するコストをどのように特定し、追跡するか。作成したリソースをユーザーが作成、管理、廃棄できるようにするメカニズムをどのように開発しますか。

質問レベルの紹介: コストメトリクスを把握するには、ソフトウェアエンジニアリング、データ管理、製品所有者、財務、リーダーシップなど、複数のチームの参加とコラボレーションが必要です。主なコスト要因を特定するには、すべての関係者がサービスの利用管理手段とコスト管理のトレードオフを把握する必要があり、多くの場合、それがコスト最適化の取り組みの成功と失敗を大きく左右します。開発から本番稼働まで、そして廃止までに作成されたリソースを追跡するプロセスとツールを用意しておく、ElastiCache に関連するコストを管理するのに役立ちます。

質問レベルのメリット: ワークロードに関連するすべてのコストを継続的に追跡するには、ElastiCache をコンポーネントの 1 つとして含むアーキテクチャを深く理解する必要があります。さらに、使用状況を収集して予算と比較するためのコスト管理計画を立てておく必要があります。

- [必須] 組織の ElastiCache の使用状況に関するメトリクスの定義、追跡、アクションの実行のために、創設憲章の 1 つとともに Cloud Center of Excellence (CCoE) を設立してください。CCoE が存在し、機能している場合は、ElastiCache に関連するコストの読み取り方法と追跡方法について把握していることを確認してください。リソースを作成したら、IAM のロールとポリシーを使用して、特定のチームとグループのみがリソースをインスタンス化できることを検証します。これにより、コストがビジネスの成果と結びつき、コストの観点から明確な説明責任が確立されます。
 1. CCoE は、次のようなカテゴリデータにおける主要な ElastiCache の使用状況について、定期的に (毎月) 更新されるコストメトリクスを特定、定義、公開する必要があります。

- a. 使用されるノードの種類とその属性: 標準インスタンスとメモリ最適化インスタンス、オンデマンドインスタンスとリザーブドインスタンス、リージョンとアベイラビリティゾーン
 - b. 環境の種類: 無料、開発、テスト、本番
 - c. バックアップストレージと保存戦略
 - d. リージョン内およびリージョン間のデータ転送
 - e. Amazon Outposts で実行されているインスタンス
2. CCoE は、組織内のソフトウェアエンジニアリング、データ管理、製品チーム、財務チーム、リーダーシップチームからの非独占的な代表者による部門横断チームで構成されています。

[リソース]:

- [Cloud Center of Excellence の作成](#)
 - [Amazon ElastiCache 料金表](#)
- [必須] コスト配分タグを使用すると、コストを低レベルの粒度で追跡できます。AWS Cost Management を使用すると、経時的な AWS のコストと使用状況を視覚化し、把握および管理できます。
1. タグを使用してリソースを整理し、コスト配分タグを使用して AWS のコストを詳細なレベルで追跡できます。コスト配分タグを有効化した後、AWS コストを分類して追跡しやすくするために、AWS はコスト配分タグを利用してコスト配分レポート上でリソースコストを整理します。AWS には 2 つのタイプのコスト配分タグ、AWS 生成のタグとユーザー定義タグがあります。AWS は AWS 生成タグを定義、作成、適用し、ユーザーはユーザー定義タグを定義、作成、適用します。Cost Management またはコスト配分レポートで使用するには、事前に両方のタイプのタグを別々にアクティブ化しておく必要があります。
 2. コスト配分タグを使用して AWS の請求書を分類し、自分のコスト構造を反映させます。Amazon ElastiCache でリソースにコスト配分タグを追加する場合、リソースのタグ値に基づいて請求書の費用をグループ化してコストを追跡できます。さらに細かくコストを追跡するには、タグを組み合わせる必要があります。

[リソース]:

- [AWS コスト配分タグを使用する](#)
 - [コスト配分タグによるコストのモニタリング](#)
 - [AWS Cost Explorer](#)
- [最良] ElastiCache のコストを組織全体に及ぶメトリクスに結び付けます。

1. ビジネスメトリクスだけでなく、レイテンシーなどの運用メトリクスも検討してください。ビジネスモデルのどの概念がロールに関係なく理解できるでしょうか。メトリクスは、組織内のできるだけ多くのロールが理解できる必要があります。
2. 例 - 同時提供ユーザー数、オペレーションとユーザーごとの最大レイテンシーと平均レイテンシー、ユーザーエンゲージメントスコア、週あたりのユーザー戻り率、ユーザーあたりのセッション時間、離脱率、キャッシュヒットレート、追跡しているキー

[リソース]:

- [CloudWatch メトリクスの使用状況のモニタリング](#)
- [良い] ElastiCache を使用するワークロード全体のメトリクスとコストについて、アーキテクチャと運用上の最新の可視性を維持します。
 1. ソリューションエコシステム全体を把握しておきます。ElastiCache は、クライアントからレポートツール用の API Gateway、Redshift、QuickSight など、自社のテクノロジーセットに含まれる AWS サービスのエコシステム全体の一部となる傾向があります。
 2. クライアント、接続、セキュリティ、インメモリオペレーション、ストレージ、リソース自動化、データアクセス、管理など、ソリューションのコンポーネントをアーキテクチャ図にマッピングします。各レイヤーはソリューション全体につながり、独自のニーズや機能を備えているため、全体的なコストの管理に追加したり、役立てることができます。
 3. 図には、コンピューティング、ネットワーク、ストレージ、ライフサイクルポリシー、メトリクス収集のほか、アプリケーションの運用上および機能上の ElastiCache 要素の使用を含める必要があります。
 4. ワークロードの要件は時間の経過とともに変化する可能性が高いため、ワークロードコスト管理を積極的に進めるためには、基礎となるコンポーネントと主要な機能目標についての理解を引き続き維持し、文書化することが不可欠です。
 5. ElastiCache の効果的なコスト管理戦略を立てるには、可視性、説明責任、優先順位付け、リソースに対するリーダーシップのサポートが不可欠です。

コスト 2: ElastiCache リソースに関連するコストの最適化のために、継続的モニタリングツールをどのように使用するか。

質問レベルの紹介: ElastiCache のコストとアプリケーションのパフォーマンスメトリクスの適切なバランスを図ることを目指す必要があります。Amazon CloudWatch は主要な運用メトリクスを可視化できるため、ElastiCache リソースの使用率が高すぎるか、または十分に活用されていないかをニーズに応じて評価できます。コスト最適化の観点からは、いつオーバープロビジョニングされている

るかを把握し、運用、可用性、回復力、パフォーマンスのニーズを維持しつつ、ElastiCache リソースのサイズを変更する適切なメカニズムを開発する必要があります。

質問レベルのメリット: 理想的な状態では、ワークロードの運用上のニーズを満たすのに十分なリソースをプロビジョニングでき、コストが最適ではない状態につながる可能性のある、十分に活用されていないリソースがないことです。サイズが大きすぎる ElastiCache リソースの長期間の使用を特定し、同時に回避できる必要があります。

- [必須] CloudWatch を使用して ElastiCache クラスターを監視し、これらのメトリクスが AWS Cost Explorer ダッシュボードとどのように関連しているかを分析します。
 1. ElastiCache では、ホストレベルのメトリクス (たとえば、CPU 使用率など) とキャッシュエンジンソフトウェアに固有のメトリクス (たとえば、キャッシュの取得やキャッシュの損失など) の両方が提供されます。これらのメトリクスは 60 秒間隔で各キャッシュノードに対して測定およびパブリッシュされます。
 2. ElastiCache のパフォーマンスメトリクス (CPUUtilization、EngineUtilization、SwapUsage、CurrConnections、Evictions) から、スケールアップ/ダウン (より大きな/小さなキャッシュノードタイプを使用) またはスケールイン/アウト (シャードの増加/減少) する必要性を示すことがあります。スケールアップに関する意思決定のコストへの影響を理解するには、追加コストと、アプリケーションパフォーマンスのしきい値を満たすために必要な最小および最大時間を推定するプレイブックマトリクスを作成します。

[リソース]:

- [CloudWatch メトリクスの使用状況のモニタリング](#)
- [モニタリングすべきメトリクス](#)
- [Amazon ElastiCache 料金表](#)
- [必須] バックアップ戦略とコストへの影響を理解し、文書化します。
 1. ElastiCache では、バックアップは耐久性に優れたストレージを提供する Amazon S3 に保存されます。障害からの復旧能力に関連するコストへの影響を理解する必要があります。
 2. 保存制限を過ぎたバックアップファイルを削除する自動バックアップを有効にします。

[リソース]:

- [自動バックアップのスケジュール](#)
- [Amazon Simple Storage Service の料金表](#)
- [最良] 十分に理解され、文書化されているワークロードのコストを管理するための計画的な戦略として、インスタンスにリザーブドノードを使用します。リザーブドノードには、ノードタイプと

予約の期間 (1 年または 3 年) に応じて、前払い料金が請求されます。この料金は、オンデマンドノードで発生する 1 時間あたりの使用料金よりもはるかに安くなります。

1. リザーブドインスタンスの要件を推定するのに十分なデータを収集するまで、オンデマンドノードを使用して ElastiCache クラスターを運用する必要がある場合があります。ニーズを満たすために必要なリソースを計画して文書化し、インスタンスタイプ (オンデマンドとリザーブド) で予想コストを比較します。
2. 利用可能な新しいキャッシュノードタイプを定期的に評価し、コストと運用メトリクスの観点から、インスタンスフリートを新しいキャッシュノードタイプに移行することが理にかなっているかどうかを評価します。

コスト 3: データ階層化をサポートするインスタンスタイプを使用すべきか。データ階層化のメリットは何か。データ階層化インスタンスを使用すべきでないのはどのような場合か。

質問レベルの紹介: 適切なインスタンスタイプを選択すると、パフォーマンスやサービスレベルだけでなく、財務面にも影響が及びます。インスタンスタイプにはそれぞれ異なるコストがかかります。そのため、メモリ内のすべてのストレージニーズに対応できる大規模なインスタンスタイプを 1 つまたはいくつか選択するのは、自然な判断かもしれません。ただし、プロジェクトが成熟するにつれて、この選択はコストに大きな影響を与える可能性があります。正しいインスタンスタイプが選択されていることを確認するには、ElastiCache オブジェクトのアイドル時間を定期的に調査する必要があります。

質問レベルのメリット: さまざまなインスタンスタイプが現在および将来のコストにどのように影響するかを明確に理解しておく必要があります。ワークロードのわずかな変化や定期的な変更によってコストが不均等に变化してはいけません。ワークロードが許せば、データ階層化をサポートするインスタンスタイプのほうが、利用可能なストレージあたりの価格が向上します。インスタンスごとに利用可能な SSD ストレージのため、データ階層化インスタンスは、インスタンス機能あたりはるかに多くの合計データ容量をサポートします。

- [必須] データ階層化インスタンスの制限を理解する

1. Redis クラスター用の ElastiCache でのみ利用可能です。
2. データ階層化をサポートしているインスタンスタイプは限られています。
3. Redis バージョン 6.2 以降の ElastiCache のみがサポートされています
4. 大きなアイテムは SSD にスワップアウトされません。128 MiB を超えるオブジェクトはメモリに保持されます。

[リソース]:

- [データ階層化](#)
- [Amazon ElastiCache 料金表](#)
- [必須] データベースの何パーセントがワークロードから定期的にアクセスされているかを把握します。
 1. データ階層化インスタンスは、データセット全体のごく一部にアクセスすることが多いものの、残りのデータへの高速アクセスが必要なワークロードに最適です。つまり、ホットデータとウォームデータの比率は約 20:80 です。
 2. オブジェクトのアイドル時間をクラスターレベルで追跡する機能を開発します。
 3. 優れた選択肢としては、500 GB を超えるデータを扱う大規模な実装を行うことです。
- [必須] 特定のワークロードでは、データ階層化インスタンスはオプションではないことを理解しておきます。
 1. 使用頻度の低いオブジェクトはローカル SSD にスワップアウトされるため、アクセス時に多少のパフォーマンスコストが発生します。アプリケーションが応答時間に敏感な場合は、ワークロードへの影響をテストしてください。
 2. 主にサイズが 128 MiB を超える大きなオブジェクトを格納するキャッシュには適していません。

[リソース]:

- [機能制限](#)
- [最良] リザーブドインスタンスタイプはデータ階層化をサポートします。これにより、インスタンスあたりのデータストレージ量の面で、コストが最小化されることが保証されます。
 1. 要件を十分に理解するまで、非データ階層化インスタンスを使用して ElastiCache クラスターを運用する必要がある場合があります。
 2. ElastiCache クラスターのデータ使用パターンを分析します。
 3. オブジェクトのアイドル時間を定期的に収集する自動ジョブを作成します。
 4. オブジェクトの大部分 (約 80%) がワークロードに適していると思われる期間アイドル状態になっていることに気付いた場合は、その結果を文書化し、データ階層化をサポートするインスタンスにクラスターを移行することを提案します。
 5. 利用可能な新しいキャッシュノードタイプを定期的に評価し、コストと運用メトリクスの観点から、インスタンスフリートを新しいキャッシュノードタイプに移行することが理にかなっているかどうかを評価します。

[リソース]:

- [OBJECT IDLETIME](#)
- [Amazon ElastiCache 料金表](#)

一般的なトラブルシューティング手順とベストプラクティス

トピック

- [接続の問題](#)
- [Redis クライアントエラー](#)
- [ElastiCache Serverless での高レイテンシーのトラブルシューティング](#)
- [ElastiCache Serverless でのスロットリングの問題のトラブルシューティング](#)
- [関連トピック](#)

接続の問題

ElastiCache キャッシュに接続できない場合は、次のいずれかを検討してください。

1. TLS の使用: ElastiCache エンドポイントに接続しようとしたときに接続がハングしている場合、クライアントで TLS を使用していない可能性があります。ElastiCache Serverless を使用している場合、転送中の暗号化は常に有効になります。クライアントが TLS を使用してキャッシュに接続していることを確認します。TLS 対応キャッシュへの接続の詳細については、[「」を参照してください](#)。
2. VPC: ElastiCache キャッシュには VPC 内からのみアクセスできます。キャッシュにアクセスしている EC2 インスタンスと ElastiCache キャッシュが同じ VPC に作成されていることを確認します。または、EC2 インスタンスが存在する VPC とキャッシュを作成する VPC との間で [VPC ピアリング](#) を有効にする必要があります。
3. セキュリティグループ: ElastiCache は、セキュリティグループを使用してキャッシュへのアクセスを制御します。以下の点を考慮します。
 - a. ElastiCache キャッシュで使用されるセキュリティグループが EC2 インスタンスからのインバウンドアクセスを許可していることを確認します。セキュリティグループでインバウンドルールを正しく設定する方法については、[こちら](#)を参照してください。
 - b. ElastiCache キャッシュで使用されるセキュリティグループがキャッシュのポート (サーバーレスの場合は 6379 および 6380、自己設計の場合は 6379) へのアクセスを許可していることを確

認めます。はこれらのポート ElastiCache を使用して Redis コマンドを受け入れます。ポートアクセスの設定方法については、[???「](#)」を参照してください。

Redis クライアントエラー

ElastiCache Serverless には、Redis クラスターモードプロトコルをサポートする Redis クライアントを使用してのみアクセスできます。自己設計のクラスターには、クラスター設定に応じて、どちらのモードでも Redis クライアントからアクセスできます。

クライアントで Redis エラーが発生した場合は、次の点を考慮してください。

1. クラスターモード: [SELECT](#) Redis コマンドで CROSSLOT エラーまたはエラーが発生した場合は、Redis クラスタープロトコルをサポートしていない Redis クライアントを使用してクラスターモード対応キャッシュにアクセスしようとしている可能性があります。ElastiCache Serverless は、Redis クラスタープロトコルをサポートする Redis クライアントのみをサポートします。「クラスターモードが無効」(CMD) で Redis を使用する場合は、独自のクラスターを設計する必要があります。
2. CROSSLOT エラー: ERR CROSSLOT Keys in request don't hash to the same slot エラーが発生した場合、クラスターモードキャッシュの同じスロットに属していないキーにアクセスしようとしている可能性があります。覚えておいてください。ElastiCache Serverless は常にクラスターモードで動作します。複数のキーを含むマルチキーオペレーション、トランザクション、または Lua スクリプトは、関連するすべてのキーが同じハッシュスロットにある場合にのみ許可されます。

Redis クライアントの設定に関するその他のベストプラクティスについては、この[ブログ記事「](#)」を参照してください。

ElastiCache Serverless での高レイテンシーのトラブルシューティング

ワークロードのレイテンシーが大きいと思われる場合は、SuccessfulReadRequestLatency および CloudWatch SuccessfulWriteRequestLatency メトリクスを分析して、レイテンシーが ElastiCache Serverless に関連しているかどうかを確認できます。これらのメトリクスは、ElastiCache サーバーレス内部のレイテンシーを測定します。クライアント側のレイテンシーとクライアントと ElastiCache サーバーレスエンドポイント間のネットワークトリップ時間は含まれません。

一部の変動や時折発生するスパイクは、懸念の原因とはなりません。ただし、Average 統計が急激に増加し、持続する場合は、AWS Health Dashboard と Personal Health Dashboard で詳細を

確認する必要があります。必要に応じて、でサポートケースを開くことを検討してください AWS Support。

レイテンシーを減らすには、次のベストプラクティスと戦略を検討してください。

- レプリカからの読み取りを有効にする: アプリケーションで許可されている場合は、Redis クライアントの「レプリカからの読み取り」機能を有効にして読み取りをスケールリングし、レイテンシーを短縮することをお勧めします。有効にすると、ElastiCache Serverless はクライアントと同じアベイラビリティゾーン (AZ) にあるレプリカキャッシュノードに読み取りリクエストをルーティングしようとします。これにより、AZ 間のネットワークレイテンシーを回避できます。クライアントでレプリカからの読み取り機能を有効にすると、アプリケーションが結果整合性をデータに受け入れることを意味します。キーへの書き込み後に読み込もうとすると、アプリケーションが古いデータを受け取ることがあります。
- アプリケーションがキャッシュと同じ AZs にデプロイされていることを確認します。アプリケーションがキャッシュと同じ AZs にデプロイされていない場合、クライアント側のレイテンシーが高くなる可能性があります。サーバーレスキャッシュを作成すると、アプリケーションがキャッシュにアクセスするサブネットを指定でき、ElastiCache サーバーレスはそれらのサブネットに VPC エンドポイントを作成します。アプリケーションが同じ AZs にデプロイされていることを確認します。そうしないと、キャッシュにアクセスするときにアプリケーションにクロス AZ ホップが発生し、クライアント側のレイテンシーが高くなる可能性があります。
- 接続の再利用: ElastiCache サーバーレスリクエストは、RESP プロトコルを使用して TLS 対応 TCP 接続を介して行われます。最初のリクエストのレイテンシーが通常よりも長くなるため、接続の開始 (設定されている場合は接続の認証を含む) には時間がかかります。既に初期化された接続を介したリクエストは、ElastiCache の一貫した低レイテンシーを実現します。このため、接続プーリングの使用または既存の Redis 接続の再利用を検討する必要があります。
- スケールリング速度: ElastiCache サーバーレスは、リクエストレートの増加に応じて自動的にスケールリングされます。ElastiCache Serverless のスケールリング速度よりも高速で、リクエストレートが急激に増加すると、レイテンシーが長くなる可能性があります。ElastiCache Serverless は通常、サポートされるリクエストレートを迅速に増やすことができ、リクエストレートが 2 倍になるまでに最大 10~12 分かかります。
- 長時間実行されるコマンドを検査する: Lua スクリプトや大規模データ構造でのコマンドなど、一部の Redis コマンドは長時間実行される可能性があります。これらのコマンドを識別するために、はコマンドレベルのメトリクス ElastiCache を発行します。[ElastiCache サーバーレス](#)では、BasedECPUs メトリクスを使用できます。
- スロットリングされたリクエスト: ElastiCache Serverless でリクエストがスロットリングされると、アプリケーションでクライアント側のレイテンシーが増加する

可能性があります。ElastiCache Serverless でリクエストがスロットリングされると、ThrottledRequests[ElastiCache Serverless](#) メトリクスが増加します。スロットリングされたリクエストのトラブルシューティングについては、以下のセクションを確認してください。

- キーとリクエストの均等分散: ElastiCache for Redis では、スロットあたりのキーまたはリクエストが不均一に分散されると、レイテンシーが高くなる可能性があります。ElastiCache Serverless は、単純な SET/GET コマンドを実行するワークロードで、1つのスロットで最大 30,000 ECPUs/秒 (レプリカからの読み取りを使用する場合は 90,000 ECPUs/秒) をサポートします。スロット間でキーとリクエストの分散を評価し、リクエストレートがこの制限を超える場合は、均一に分散させることをお勧めします。

ElastiCache Serverless でのスロットリングの問題のトラブルシューティング

サービス指向アーキテクチャや分散システムでは、API 呼び出しが各種サービスコンポーネントによって処理される速度を制限することをスロットリングと呼びます。これにより、コンポーネントのスループットの不一致のスパイク、コントロールがスムーズになり、予期しない運用イベントが発生した場合に、より予測可能な復旧が可能になります。ElastiCache Serverless はこれらのタイプのアーキテクチャ向けに設計されており、ほとんどの Redis クライアントには、スロットリングされたリクエストの再試行が組み込まれています。ある程度のスロットリングはアプリケーションにとって必ずしも問題とはなりません。データワークフローのレイテンシーの影響を受けやすい部分を継続的にスロットリングすると、ユーザーエクスペリエンスに悪影響を及ぼし、システム全体の効率を低下させる可能性があります。

ElastiCache Serverless でリクエストがスロットリングされる

と、ThrottledRequests[ElastiCache Serverless](#) メトリクスが増加します。スロットリングされたリクエストの数が多いことに気付いた場合は、次の点を考慮してください。

- スケーリング速度: ElastiCache サーバーレスは、データの取り込みやリクエストレートの増加に伴って自動的にスケーリングされます。アプリケーションが ElastiCache サーバーレスのスケーリング速度よりも速くスケールすると、ワークロードに合わせて ElastiCache サーバーレスがスケールしている間にリクエストがスロットリングされることがあります。通常、ElastiCache サーバーレスはストレージサイズを迅速に増やすことができ、キャッシュ内のストレージサイズの 2 倍に最大 10~12 分かかります。
- キーとリクエストの均等分散: ElastiCache for Redis では、スロットあたりのキーまたはリクエストの分散が不均一になると、ホットスロットになる可能性があります。1つのスロットへのリクエ

ストレートが、単純な SET/GET コマンドを実行するワークロードで 30,000 ECPUs/秒を超えると、ホットスポットによってリクエストのスポットリングが発生する可能性があります。

- レプリカからの読み取り: アプリケーションで許可されている場合は、「レプリカからの読み取り」機能の使用を検討してください。ほとんどの Redis クライアントは、読み取りをレプリカノードに転送するように「読み取りのスケーリング」を設定できます。この機能により、読み取りトラフィックをスケーリングできます。さらに、ElastiCache サーバーレスは、アプリケーションと同じアベイラビリティーゾーンのノードにレプリカリクエストからの読み取りを自動的にルーティングするため、レイテンシーが短縮されます。レプリカからの読み取りを有効にすると、単純な SET/GET コマンドを使用するワークロードに対して、1つのスポットで最大 90,000 ECPUs/秒を実現できます。

関連トピック

- [その他のトラブルシューティング手順](#)
- [the section called “ベストプラクティスとキャッシュ戦略”](#)

その他のトラブルシューティング手順

を使用して持続的な接続問題をトラブルシューティングする際には、次の項目を確認する必要があります ElastiCache。

トピック

- [セキュリティグループ](#)
- [ネットワーク ACL](#)
- [ルートテーブル](#)
- [DNS 解決](#)
- [サーバー側の診断に関する問題の特定](#)
- [ネットワーク接続性の検証](#)
- [ネットワーク関連の制限](#)
- [CPU 使用率](#)
- [サーバー側からの接続が終了している](#)
- [Amazon EC2 インスタンスのクライアント側のトラブルシューティング](#)
- [1つのリクエストを完了するのにかかった時間の解読](#)

セキュリティグループ

セキュリティグループは、ElastiCache クライアント (EC2 インスタンス、AWS Lambda 関数、Amazon ECS コンテナなど) ElastiCache とキャッシュを保護する仮想ファイアウォールです。セキュリティグループはステートフルです。つまり、着信トラフィックまたは発信トラフィックが許可されると、そのトラフィックに対する応答は、その特定のセキュリティグループのコンテキストで自動的に承認されます。

ステートフル機能では、セキュリティグループがすべての許可された接続を追跡し続ける必要があります。追跡される接続には制限があります。制限に到達すると、新しい接続は失敗します。ElastiCache クライアント側または側で制限に達したかどうかを確認する方法については、トラブルシューティングセクションを参照してください。

ElastiCache クライアントとクラスターに同時に 1 つのセキュリティグループを割り当てることも、それぞれに個別のセキュリティグループを割り当てることもできます。

いずれの場合も、送信元からのポートの TCP アウトバウンドトラフィックと、ElastiCache 同じポートからへのインバウンドトラフィックを許可する必要があります。ElastiCache デフォルトのポートは、Memcached では 11211、Redis では 6379 です。デフォルトで、セキュリティグループはすべてのアウトバウンドトラフィックを許可します。この場合、ターゲットセキュリティグループのインバウンドルールのみが必要です。

詳細については、「[Amazon VPC ElastiCache のクラスターにアクセスするためのアクセスパターン](#)」を参照してください。

ネットワーク ACL

ネットワークアクセスコントロールリスト (ACL) は、ステートレスルールです。トラフィックを成功させるには、両方向 (インバウンドとアウトバウンド) で許可する必要があります。ネットワーク ACL は、特定のリソースではなく、サブネットに割り当てられます。クライアントリソースに同じ ACL ElastiCache を割り当てることは可能です。特に、それらが同じサブネット内にある場合はそうです。

デフォルトでは、ネットワーク ACL はすべてのトラフィックを許可します。ただし、トラフィックを拒否または許可するようにカスタマイズすることは可能です。さらに、ACL ルールの評価はシーケンシャルです。つまり、トラフィックに一致する最も小さい番号を持つルールで、それが許可または拒否されます。Redis トラフィックを許可する最小構成は次のとおりです。

クライアント側のネットワーク ACL:

- インバウンドルール:
- ルール番号: 好ましくは拒否ルールよりも低い。
- [タイプ]: [カスタム TCP ルール]。
- [Protocol]: TCP
- [Port Range]: 1024 ~ 65535
- ソース:0.0.0.0/0 (またはクラスターサブネットの個別のルールを作成) ElastiCache
- [許可/拒否]: 許可

- アウトバウンドルール:
- ルール番号: 好ましくは拒否ルールよりも低い。
- [タイプ]: [カスタム TCP ルール]。
- [Protocol]: TCP
- [Port Range]: 6379
- ソース:0.0.0.0/0 (またはクラスターサブネット。ElastiCache 特定の IP を使用すると、フェイルオーバーやクラスターのスケーリング時に問題が発生する可能性があることに注意してください)
- [許可/拒否]: 許可

ElastiCache ネットワーク ACL:

- インバウンドルール:
- ルール番号: 好ましくは拒否ルールよりも低い。
- [タイプ]: [カスタム TCP ルール]。
- [Protocol]: TCP
- [Port Range]: 6379
- ソース:0.0.0.0/0 (またはクラスターサブネットの個別のルールを作成) ElastiCache
- [許可/拒否]: 許可

- アウトバウンドルール:
- ルール番号: 好ましくは拒否ルールよりも低い。
- [タイプ]: [カスタム TCP ルール]。
- [Protocol]: TCP

- [Port Range]: 1024 ~ 65535
- ソース:0.0.0.0/0 (またはクラスターサブネット。ElastiCache 特定の IP を使用すると、フェイルオーバーやクラスターのスケールリング時に問題が発生する可能性があることに注意してください)
- [許可/拒否]: 許可

詳細については、「[ネットワーク ACL](#)」を参照してください。

ルートテーブル

ネットワーク ACL と同様に、各サブネットは異なるルートテーブルを持つことができます。ElastiCache クライアントとクラスターが異なるサブネットにある場合は、それぞれのルートテーブルが相互にアクセスできるようにしてください。

複数の VPC、動的ルーティング、またはネットワークファイアウォールを含む、より複雑な環境は、トラブルシューティングが困難になることがあります。「[ネットワーク接続性の検証](#)」を参照して、ネットワーク設定が適切であることを確認してください。

DNS 解決

ElastiCache DNS 名に基づいてサービスエンドポイントを提供します。使用可能なエンドポイントは、Configuration、Primary、Reader、および Node エンドポイントです。詳細情報については、「[接続エンドポイントの検索](#)」を参照してください。

フェイルオーバーまたはクラスターの変更の場合、エンドポイント名に関連付けられたアドレスが変更され、自動的に更新されます。

カスタム DNS 設定 (つまり、VPC DNS サービスを使用しない) では、ElastiCache 提供された DNS 名が認識されない場合があります。dig(以下に示すように) やなどのシステムツールを使用して、ElastiCache システムがエンドポイントを正常に解決できることを確認してください。nslookup

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com
example-001.xxxxxx.0001.use1.cache.amazonaws.com.
1.2.3.4
```

VPC DNS サービスを介して名前解決を強制することもできます。

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com @169.254.169.253
example-001.tihewd.0001.use1.cache.amazonaws.com.
```

サーバー側の診断に関する問題の特定

CloudWatch ElastiCache エンジンから得られるメトリクスとランタイム情報は、接続の問題の潜在的な原因を特定するための一般的な情報源または情報です。適切な分析は、通常、次の項目から始まります。

- CPU使用率: Redis はマルチスレッドアプリケーションです。ただし、各コマンドの実行は単一の (メイン) スレッドで行われます。このため、ElastiCache CPUUtilizationEngineCPUUtilizationはメトリクスとを提供しています。EngineCPUUtilizationRedis プロセス専用の CPU 使用率と、すべての CPUUtilization vCPUs の使用率を示します。複数の vCPU を持つノードは、通常、CPUUtilization と EngineCPUUtilization で異なる値を持ち、ここで 2 番目が一般的に高くなります。高い EngineCPUUtilization は、リクエスト数の上昇や、完了までに多くの CPU 時間を要する複雑なオペレーションが原因である可能性があります。両方を特定するには、以下を使用します。
- リクエスト数の上昇: EngineCPUUtilization パターンに一致する他のメトリクスでの上昇がないか確認します。役に立つメトリクスは次のとおりです。
- CacheHits と CacheMisses: 成功したリクエスト数、またはキャッシュで有効な項目を見つけれなかったリクエスト数。ヒットに対するミスの比率が高い場合、アプリケーションは実りのないリクエストで時間とリソースを浪費しています。
- SetTypeCmds と GetTypeCmds: EngineCPUUtilization と関連しているこれらのメトリクスは、SetTypeCmds によって測定された書き込みリクエスト、または GetTypeCmds によって測定された読み取りに対して負荷が著しく高いかどうかを理解するのに役立ちます。負荷が主に読み取りである場合、複数のリードレプリカを使用すると、複数のノード間でリクエストをバランスさせ、プライマリを書き込み用にとっておくことができます。クラスターモードが無効になっているクラスターでは、リーダーエンドポイントを使用してアプリケーションに追加の接続設定を作成することで、リードレプリカを使用できます。ElastiCache 詳細情報については、「[接続エンドポイントの検索](#)」を参照してください。読み取りオペレーションは、この追加の接続に送信する必要があります。書き込みオペレーションは、通常のプライマリエンドポイントを介して実行されます。クラスターモードが有効の場合、リードレプリカをネイティブにサポートするライブラリを使用することをお勧めします。適切なフラグを使用すると、ライブラリはクラスタポート、レプリカノードを自動的に検出し、[READONLY](#) Redis コマンドを介して読み取りオペレーションを有効にし、読み取りリクエストをレプリカに送信します。
- 昇格された接続数:

- `CurrConnections` と `NewConnections`: `CurrConnection` はデータポイント収集時に確立された接続の数であり、`NewConnections` はその期間に作成された接続数を示します。

接続の作成と処理は、多くの CPU オーバーヘッドの発生を意味します。さらに、新しい接続の作成に必要な TCP スリーウェイハンドシェイクは、全体的な応答時間に悪影響を及ぼします。

`NewConnections` 1 ElastiCache 分あたり数千のノードは、ほんの数個のコマンドで接続が作成され使用されていることを意味しますが、これは最適ではありません。確立された接続を維持し、新しいオペレーションのために接続を再使用することがベストプラクティスです。これは、クライアントアプリケーションが接続プーリングまたは永続的な接続をサポートし、適切に実装している場合に可能です。接続プーリングでは、`currConnections` の数には大きなバリエーションがないので、`NewConnections` をできる限り低くする必要があります。Redis は、少数の `CurrConnections` で最適なパフォーマンスを提供します。`CurrConnection` を数十または数百のオーダーで維持すると、クライアントバッファのような個別の接続や、接続に役立つ CPU サイクルをサポートするためのリソースの使用量が最小限になります。

- ネットワークスループット:

- 帯域幅の決定: ElastiCache ノードのネットワーク帯域幅はノードのサイズに比例します。アプリケーションの特性が異なるため、結果はワークロードに応じて異なる場合があります。たとえば、小さなリクエストの割合が高いアプリケーションは、ネットワークのスループットよりも CPU 使用率に影響を及ぼす傾向がありますが、キーが大きいほどネットワーク使用率が高くなります。そのため、制限をよりよく理解するために、実際のワークロードでノードをテストすることをお勧めします。

アプリケーションからの負荷をシミュレートすると、より正確な結果が得られます。ただし、ベンチマークツールは、制限についての良いアイデアを提供することができます。

- リクエストが主に読み取りの場合、読み取りオペレーションでレプリカを使用すると、プライマリノードの負荷が軽減されます。ユースケースが主に書き込みの場合、多くのレプリカを使用するとネットワークの使用が増大します。プライマリノードに書き込まれるすべてのバイトについて、N バイトがレプリカに送信され、ここで N はレプリカの数になります。書き込みの多いワークロードのベストプラクティスは、複数のシャード間で書き込みを分散できるようにクラスターモードを有効にした Redis ElastiCache を使用するが、より多くのネットワーク機能を備えたノードタイプにスケールアップすることです。
- `CloudWatchmetrics NetworkBytesIn` とは、`NetworkBytesOut` ノードに出入りするデータ量をそれぞれ提供します。`ReplicationBytes` はデータ複製専用のトラフィックです。

詳細については、「[ネットワーク関連の制限](#)」を参照してください。

- 複雑なコマンド: Redis コマンドは単一のスレッドで提供されます。つまり、リクエストは順番に処理されます。単一のスローコマンドは、他のリクエストや接続に影響を及ぼし、タイムアウトが発生する可能性があります。複数の値、キー、またはデータ型に作用するコマンドの使用は、慎重に行う必要があります。接続は、パラメータの数や入出力値のサイズに応じて、ブロックまたは終了できます。

評判の悪い例は、KEYS コマンドです。これは、指定されたパターンを検索するキースペース全体をスイープし、その実行中に他のコマンドの実行をブロックします。Redis は、コマンドの複雑さを記述するために「Big O」表記法を使用しています。

キーコマンドには $O(N)$ 時間の複雑さがあり、ここで N はデータベース内のキーの数になります。したがって、キーの数が多いほど、コマンドは遅くなります。KEYS は、さまざまな方法で問題を引き起こす可能性があります。検索パターンが使用されていない場合、コマンドは利用可能なすべてのキー名を返します。数千または数百万の項目を含むデータベースでは、巨大な出力が作成され、ネットワークバッファがフラッシングします。

検索パターンを使用すると、パターンに一致するキーのみがクライアントに戻ります。ただし、エンジンはキースペース全体のスイープを続けるので、コマンドを完了する時間は同じになります。

KEYS の代替は、SCAN コマンドです。これは、キースペースを反復し、特定の数の項目の反復を制限して、エンジン上の長引くブロックを回避します。

スキャンには COUNT パラメータがあり、反復ブロックのサイズを設定するために使用されます。デフォルト値は 10 (1 回の反復あたり 10 個のアイテム) です。

データベース内の項目数に応じて、小さな COUNT 値のブロックは、フルスキャンを完了するために多くの反復を必要とし、値を大きくすると、各反復でエンジンをビジー状態でより長く維持します。小さなカウント値は大きなデータベースで SCAN をより遅くしますが、値を大きくすると KEYS で説明されたものと同じ問題が生じる可能性があります。

例として、10 のカウント値がある SCAN コマンドの実行は、100 万個のキーがあるデータベースでの 100,000 回の繰り返しの必要とします。平均ネットワークラウンドトリップ時間が 0.5 ミリ秒の場合、リクエストの転送に約 50,000 ミリ秒 (50 秒) が費やされます。

一方、カウント値が 100,000 であった場合、1 回の反復が必要で、転送に費やされる時間はわずか 0.5 ミリ秒です。ただし、コマンドがすべてのキースペースのスweepを終了するまで、エンジンは他のオペレーションのために完全にブロックされます。

KEYSに加えて、いくつかの他のコマンドは、正しく使用されない場合、潜在的に有害になります。すべてのコマンドのリストとそれぞれの時間の複雑さを確認するには、<https://redis.io/commands> にアクセスしてください。

潜在的な問題の例:

- **Lua スクリプト:** Redis は組み込み Lua インタプリタを提供し、サーバー側でのスクリプトの実行を可能にします。Redis の Lua スクリプトはエンジンレベルで実行され、定義上アトミックです。つまり、スクリプトの実行中に他のコマンドやスクリプトは実行できません。Lua スクリプトは、複数のコマンド、意思決定アルゴリズム、データ解析などを Redis エンジン上で直接実行する可能性を提供します。スクリプトのアトミック性とアプリケーションのオフロードの機会は魅力的ですが、スクリプトは小さなオペレーションでは注意を払って使用する必要があります。オン ElastiCache、Lua スクリプトの実行時間は 5 秒に制限されています。キースペースに書き込まれていないスクリプトは、5 秒後に自動的に終了します。データの破損や不整合を避けるため、スクリプトの実行が 5 秒以内に完了せず、実行中に書き込みがあった場合、ノードはフェイルオーバーします。[\[トランザクション\]](#) は、Redis の複数の関連するキー変更の一貫性を保証する代替手段です。トランザクションは、コマンドのブロックの実行を可能にし、既存のキーの変更を監視します。トランザクションの完了前に監視キーのいずれかが変更された場合、すべての変更は破棄されます。
- **アイテムの一括削除:** DEL コマンドは、削除するキー名である複数のパラメータを受け入れます。削除オペレーションは同期的であり、パラメータのリストが大きい場合、または大きなリスト、セット、ソートされたセット、またはハッシュ (いくつかのサブ項目を保持するデータ構造) が含まれている場合、多くの CPU 時間がかかります。言い換えれば、単一のキーを削除しても、多くの要素がある場合、多くの時間がかかることがあります。DEL の代替は UNLINK です。これは、Redis 4 以降で利用可能な非同期コマンドです。いつでも可能な限り、DEL が UNLINK より優先される必要があります。Redis 6x 以降では、ElastiCache lazyfree-lazy-user-delDELこのパラメーターによりコマンドが有効になったときと同様に動作するようになります。UNLINK詳細については、[Redis 6.0 パラメータの変更](#)を参照してください。
- **複数のキーに作用するコマンド:** DEL は、複数の引数を受け入れるコマンドとして前述されており、実行時間はそれに正比例します。ただし、Redis には、同様に機能する多くのコマンドが用意されています。例として、MSET と MGET を使用すると、一度に複数の String キーを挿入または取得できます。これらの使用は、複数の個別の SET または GET コマンドに固有の

ネットワーク遅延を低減するために有益である可能性があります。ただし、パラメータの広範なリストは CPU 使用率に影響します。

CPU 使用率だけが接続の問題の原因ではありませんが、複数のキーで単一または少数のコマンドを処理するのに時間がかかりすぎると、他のリクエストが失敗し、CPU 使用率が増加する可能性があります。

キーの数とそのサイズは、コマンドの複雑さ、また結果として完了時間に影響します。

複数のキーに対して作用できるコマンドのその他の例:

HMGET、HMSET、MSETNX、PFCOUNT、PFMERGE、SDIFF、SDIFFSTORE、SINTER、SINTERSTORE
または ZINTERSTORE。

- 複数のデータ型に作用するコマンド: Redis はまた、そのデータ型に関係なく、1 つまたは複数のキーに作用するコマンドを提供します。ElastiCache for Redis には、KeyBasedCmds このようなコマンドを監視するためのメトリクスが用意されています。このメトリクスは、選択した期間における次のコマンドの実行を合計します。
 - $O(N)$ の複雑さ:
 - KEYS
 - $O(1)$
 - EXISTS
 - OBJECT
 - PTTL
 - RANDOMKEY
 - TTL
 - TYPE
 - EXPIRE
 - EXPIREAT
 - MOVE
 - PERSIST
 - PEXPIRE
 - PEXPIREAT
 - UNLINK ($O(N)$ により、メモリを再利用します。しかし、メモリ再利用タスクは分離されたスレッドで発生し、エンジンをブロックしません

- データ型によって異なる複雑さの時間:

- DEL
- DUMP
- RENAME は $O(1)$ の複雑さがあるコマンドとみなされますが、DEL を内部で実行します。実行時間は、名前が変更されたキーのサイズによって異なります。
- RENAMENX
- RESTORE
- SORT
- 大きなハッシュ: ハッシュは、複数のキー値サブ項目がある単一のキーを可能にするデータ型です。各ハッシュは 4.294.967.295 項目を格納することができ、大きなハッシュでのオペレーションは費用がかかる可能性があります。KEYS と同様に、ハッシュは $O(N)$ 時間の複雑さがある HKEYS コマンドを持ち、ここで N はハッシュ内の項目数になります。長時間実行されるコマンドを避けるために、HSCAN が HKEYS より優先される必要があります。HDEL、HGETALL、HMGET、HMSET および HVALS は、大きなハッシュで注意して使用する必要があるコマンドです。
- 他の大きなデータ構造: ハッシュに加えて、他のデータ構造では CPU 集中的になる可能性があります。セット、リスト、ソートされたセット、およびハイパーログログも、そのサイズと使用されるコマンドによっては、操作に多くの時間がかかることがあります。これらのコマンドの詳細については、「<https://redis.io/commands>」を参照してください。

ネットワーク接続性の検証

DNS 解決、セキュリティグループ、ネットワーク ACL、およびルートテーブルに関連するネットワーク設定を確認した後、VPC Reachability Analyzer とシステムツールを使用して接続性を検証できます。

Reachability Analyzer は、ネットワーク接続性をテストし、すべての要件と許可が満たされているかどうかを確認します。以下のテストでは、VPC ElastiCache で使用可能なノードの 1 つの ENI ID (Elastic Network Interface ID) が必要です。それを見つけるには、以下の操作を行います。

1. <https://console.aws.amazon.com/ec2/v2/home?#NIC:> にアクセスします。
2. ElastiCache クラスター名または以前に DNS 検証で取得した IP アドレスでインターフェースリストをフィルタリングします。
3. ENI ID を書き留めるか、保存してください。複数のインターフェースが表示される場合は、ElastiCache 説明を見直して正しいクラスターに属していることを確認し、そのうちの 1 つを選択してください。

4. 次のステップに進みます。
5. <https://console.aws.amazon.com/vpc/home?#ReachabilityAnalyzer> で分析パスを作成し、以下のオプションを選択します。
 - ソースタイプ: ElastiCache クライアントが Amazon EC2 インスタンスで実行されている場合はインスタンスを選択し、別のサービス (たとえば AWS Fargate Amazon ECS with awsvpc network など) を使用する場合はネットワークインターフェイスを選択し AWS Lambda、それぞれのリソース ID (EC2 インスタンスまたは ENI ID) を指定します。
 - [送信先タイプ]: [ネットワークインターフェイス] を選択し、リストから [ElastiCache ENI] を選択します。
 - 送信先ポート: Redis の場合は 6379 を指定し、Memcached ElastiCache の場合は 11211 を指定します。ElastiCache これらはデフォルト設定で定義されたポートであり、この例では、変更されていないことを前提としています。
 - [Protocol]: TCP

分析パスを作成し、結果まで数分待ちます。ステータスが到達不能の場合は、解析の詳細を開き、[解析エクスプローラ] で、リクエストがブロックされた場所の詳細を確認してください。

到達可能性テストに合格した場合は、OS レベルでの検証に進みます。

ElastiCache サービスポートの TCP 接続を検証するには: Amazon Linux では、Npingnmapパッケージに含まれており、ElastiCache ポートの TCP 接続をテストできるほか、接続を確立するためのネットワーク往復時間も表示されます。これを使用して、以下に示すように、ElastiCache ネットワーク接続とクラスターへの現在のレイテンシーを検証します。

```
$ sudo nping --tcp -p 6379 example.xxxxxx.ng.0001.use1.cache.amazonaws.com
```

```
Starting Nping 0.6.40 ( http://nmap.org/nping ) at 2020-12-30 16:48 UTC  
SENT (0.0495s) TCP ...  
(Output suppressed )
```

```
Max rtt: 0.937ms | Min rtt: 0.318ms | Avg rtt: 0.449ms  
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)  
Nping done: 1 IP address pinged in 4.08 seconds
```

デフォルトでは、nping は、5 つのプロブをそれらの間で 1 秒の遅延で送信します。オプション「-c」を使用してプロブ数を増やし、「—delay」を使用して新しいテストを送信するための時間を変更できます。

ping でのテストが失敗し、[VPC Reachability Analyzer] テストに合格した場合は、オペレーティングシステムレベルで考えられる、ホストベースのファイアウォールルール、非対称ルーティングルール、またはその他の制限を確認するよう、システム管理者に依頼してください。

ElastiCache コンソールで、ElastiCache クラスターの詳細で [転送中の暗号化] が有効になっているかどうかを確認します。転送中の暗号化が有効になっている場合は、次のコマンドを使用して TLS セッションを確立できるかどうかを確認します。

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

接続と TLS ネゴシエーションが成功すると、広範な出力が期待されます。最後の行で利用可能な戻りコードを確認してください。値は 0 (ok) である必要があります。openssl が何か異なるものを返す場合は、<https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS> でエラーの理由を確認します。

インフラストラクチャーとオペレーティング・システムのテストはすべて合格しても、アプリケーションが接続できない場合は ElastiCache、ElastiCache アプリケーションの構成が設定に準拠しているかどうかを確認してください。よくある間違いは次のとおりです。

- アプリケーションがクラスターモードをサポートしておらず ElastiCache、ElastiCache クラスターモードが有効になっている。
- アプリケーションが TLS/SSL をサポートしておらず ElastiCache、転送中の暗号化が有効になっている。
- アプリケーションは TLS/SSL をサポートしていますが、正しい設定フラグまたは信頼できる証明権限がありません。

ネットワーク関連の制限

- 最大接続数: 同時接続にはハード制限があります。ElastiCache 各ノードでは、すべてのクライアントで最大 65,000 の同時接続が可能です。この制限は、CurrConnections CloudWatch のメトリックで監視できます。ただし、クライアントにはアウトバウンド接続にも制限があります。Linux では、次のコマンドを使用して、許可されているエフェメラルポート範囲を確認してください。

```
# sysctl net.ipv4.ip_local_port_range  
net.ipv4.ip_local_port_range = 32768 60999
```

前の例では、同じ送信元、同じ宛先 IP (ElastiCache ノード)、ポートへの 28231 件の接続が許可されます。以下のコマンドは、ElastiCache 特定のノード (IP 1.2.3.4) に存在する接続数を示します。

```
ss --numeric --tcp state connected "dst 1.2.3.4 and dport == 6379" | grep -vE  
'^State' | wc -l
```

この数値が大きすぎると、接続リクエストを処理しようとしてシステムが過負荷になることがあります。接続をより適切に処理するために、接続プーリングや永続的な接続などの技術の実装を検討することをお勧めします。いつでも可能な限り、接続プールを設定して、接続の最大数を数百に制限します。また、タイムアウトやその他の接続例外を処理するためのバックオフロジックは、問題が発生した場合に接続チャーンを避けるようにすることをお勧めします。

- ネットワークトラフィック制限:[Redis CloudWatch の以下のメトリクスを確認して](#)、ノードで発生する可能性のあるネットワーク制限を特定してください。ElastiCache
- NetworkBandwidthInAllowanceExceeded/NetworkBandwidthOutAllowanceExceeded: スループットが集約された帯域幅制限を超えたためにシェーピングされたネットワークパケット。

プライマリノードに書き込まれるすべてのバイトが N 個のレプリカに複製されることに注意することが重要で、ここで N はレプリカの数になります。小さなノードタイプ、複数のレプリカ、および集中的な書き込みリクエストがあるクラスターは、レプリケーションのバックログに対処できない場合があります。このような場合は、スケールアップ (ノードタイプを変更する)、スケールアウト (クラスターモードが有効なクラスターにシャードを追加する)、レプリカ数を減らす、または書き込み数を最小限に抑えることがベストプラクティスです。

- NetworkConntrackAllowanceExceeded: ノードに割り当てられたすべてのセキュリティグループで追跡される接続の最大数を超過したため、パケットがシェーピングされます。この期間中、新しい接続が失敗する可能性があります。
- NetworkPackets PerSecondAllowanceExceeded: 1 秒あたりの最大パケット数を超過しています。非常に小さなリクエストの高いレートに基づくワークロードは、最大帯域幅よりも前にこの制限にヒットした可能性があります。

上記のメトリクスは、ノードがネットワーク制限にヒットしていることを確認するための理想的な方法です。ただし、制限はネットワークメトリクスのプラトーによっても特定できます。

プラトーが長期間にわたって観察される場合、レプリケーションの遅れ、キャッシュで使用されるバイト数の増加、解放可能なメモリの低下、高いスワップおよび CPU 使用率がそれに続きます。

また、Amazon EC2 インスタンスには、[\[ENA ドライバーメトリクス\]](#) を介して追跡できるネットワーク制限があります。拡張ネットワークングサポートおよび ENA ドライバー 2.2.10 以降を備えた Linux インスタンスは、次のコマンドを使用して制限カウンターを確認できます。

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU 使用率

CPU 使用率メトリクスは調査の出発点です。以下の項目は、発生する可能性のある問題を絞り込むのに役立ちます ElastiCache。

- Redis SlowLogs: ElastiCache 既定の構成では、完了までに 10 ミリ秒以上かかった最後の 128 個のコマンドが保持されます。スローコマンドの履歴は、エンジンランタイム中は保持され、障害や再起動時に失われます。リストが 128 エントリに達すると、古いイベントは削除され、新しいイベントのためのスペースが開きます。スローイベントのリストとスローとみなされる実行時間のサイズは、[\[カスタムパラメータグループ\]](#) のパラメータ `slowlog-max-len` および `slowlog-log-slower-than` を介して変更できます。スローログのリストは、エンジンで `SLOWLOG GET 128` を実行して取得でき、ここで 128 は最後に報告された 128 のスローコマンドになります。各エントリには以下のフィールドがあります。

```
1) 1) (integer) 1 -----> Sequential ID
   2) (integer) 1609010767 --> Timestamp (Unix epoch time)of the Event
   3) (integer) 4823378 -----> Time in microseconds to complete the command.
   4) 1) "keys" -----> Command
      2) "*" -----> Arguments
   5) "1.2.3.4:57004"-> Source
```

上記のイベントは 12 月 26 日 19:26:07 (UTC) に起こり、完了までに 4.8 秒 (4.823ms) かかり、クライアント 1.2.3.4 からリクエストされた KEYS コマンドによって発生しました。

Linux では、タイムスタンプはコマンド `date` で変換できます。

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

Python の場合:

```
>>> from datetime import datetime
```

```
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

または、Windows では以下のコマンドを実行することもできます。 PowerShell

```
PS D:\Users\user> [datetimeoffset]::FromUnixTimeSeconds('1609010767')
DateTime           : 12/26/2020 7:26:07 PM
UtcDateTime        : 12/26/2020 7:26:07 PM
LocalDateTime      : 12/26/2020 2:26:07 PM
Date               : 12/26/2020 12:00:00 AM
Day               : 26
DayOfWeek          : Saturday
DayOfYear          : 361
Hour              : 19
Millisecond        : 0
Minute            : 26
Month             : 12
Offset            : 00:00:00Ticks           : 637446075670000000
UtcTicks          : 637446075670000000
TimeOfDay         : 19:26:07
Year              : 2020
```

短時間 (同じ分以下) での多くのスローコマンドは、懸念の理由になります。コマンドの性質と、それらを最適化する方法を確認してください (前の例を参照)。O(1) の時間の複雑さがあるコマンドが頻繁に報告される場合は、前述の CPU 使用率が高いかどうかについて他の要因を確認してください。

- レイテンシーメトリクス: ElastiCache Redis には、CloudWatch さまざまなクラスのコマンドの平均レイテンシーを監視するメトリクスが用意されています。データポイントは、カテゴリ内のコマンドの実行総数を期間内の合計実行時間で割って計算されます。レイテンシーメトリクスの結果は、複数のコマンドの集合であることを理解することが重要です。1つのコマンドで、メトリクスに大きな影響を与えずに、タイムアウトのような予期しない結果が発生する可能性があります。このような場合、スローログイベントはより正確な情報源になります。次のリストには、使用可能なレイテンシーメトリクスと、それらに影響する各コマンドが含まれています。
- EvalBasedCmdsLatency: Lua スクリプトコマンドに関連する、`eval` `evalsha`

- **GeoSpatialBasedCmdsLatency:** geodist, geohash, geopos, georadius, georadiusbymember, geoadd;
- **GetTypeCmdsLatency:** データ型に関係なくコマンドを読み込む。
- **HashBasedCmdsLatency:** hexists, hget, hgetall, hkeys, hlen, hmget, hvals, hstrlen, hdel, hincrby, hincrbyfloat, hmset, hset, hsetnx;
- **HyperLogLogBasedCmdsLatency:** pfselftest, pfcount, pfdebug, pfadd, pfmerge;
- **KeyBasedCmdsLatency:** さまざまなデータタイプを処理できるコマンド: dump,,,exists,keys,object,pttl,randomkey,ttl,type,del,expire,expireat,move,persist
- **ListBasedCmdsLatency:** インデックス、レン、レンジ、プロップ、プロップ、プロップ、プロッププッシュ、インサート、ループ、プッシュ、プッシュユックス、レム、レット、ルトリム、ロップ、プロップ、プロッププッシュ、プッシュ、rpushx;
- **PubSubBasedCmdsLatency:** 購読、公開、公開/購読、購読解除、購読、購読解除
- **SetBasedCmdsLatency:** scard, sdiff, sinter, sismember, smembers, srandmember, sunion, sadd, sdiffstore, sinterstore, smove, spop, srem, sunionstore;
- **SetTypeCmdsLatency:** データ型に関係なくコマンドを書き込みます。
- **SortedSetBasedCmdsLatency:** zcard, zcount, zrange, zrangebyscore, zrank, zrevrange, zrevrangebyscore, zrevrank, zscore, zrangebylex, zrevrangebylex, zlexcount, zadd, zincrby, zinterstore, zrem, zremrangebyrank, zremrangebyscore, zunionstore, zremrangebylex, zpopmax, zpopmin, bzpopmin, bzpopmax;
- **StringBasedCmdsLatency:** bitcount, get, getbit, getrange, mget, strlen, substr, bitpos, append, bitop, bitfield, decr, decrby, getset, incr, incrby, incrbyfloat, mset, msetnx, psetex, set, setbit, setex, setnx, setrange;
- **StreamBasedCmdsLatency:** xrange, xrevrange, xlen, xread, xpending, xinfo, xadd, xgroup, readgroup, xack, xclaim, xdel, xtrim, xsetid;
- **Redis ランタイムコマンド:**
 - **info commandstats:** Redis エンジンの起動後に実行されたコマンドのリスト、それらの累積実行数、合計実行時間、およびコマンドごとの平均実行時間を提供します。
 - **client list:** 現在接続されているクライアントのリスト、およびバッファの使用状況、最後に実行されたコマンドなどの関連情報を提供します。
- **Backup とレプリケーション:** ElastiCache 2.8.22 より前の Redis バージョンでは、フォークプロセスを使用してバックアップを作成し、レプリカとの完全同期を処理します。このメソッドは、書き込み集中的なユースケースのために多くのメモリオーバーヘッドが発生する可能性があります。

ElastiCache Redis 2.8.22 から、フォークレスバックアップとレプリケーションの方法が導入されました。AWS 新しい方法は、障害を防ぐために書き込みを遅らせる場合があります。どちらの方法でも、CPU 使用率が高い期間が発生し、応答時間が長くなり、その結果、実行中にクライアントのタイムアウトが発生する可能性があります。バックアップウィンドウの間にクライアントの障害が発生したか、または SaveInProgress メトリクスが期間内で 1 であったかどうかを常に確認してください。クライアントの問題やバックアップ障害の可能性を最小限にするために、使用率の低い期間でバックアップウィンドウをスケジュールすることをお勧めします。

サーバー側からの接続が終了している

Redis 設定のデフォルトでは ElastiCache、クライアント接続は無期限に確立されたままになります。ただし、状況によっては、接続の終了が望ましい場合があります。例:

- クライアントアプリケーションのバグにより、接続が忘れられ、アイドル状態で確立されたままになることがあります。これは「接続リーク」と呼ばれ、その結果は CurrConnections メトリクスで観測される確立された接続の数の着実な増加となります。この動作により、クライアント側または側が飽和状態になる可能性があります。ElastiCache クライアント側からすぐに修正できない場合は、ElastiCache パラメータグループに「タイムアウト」値を設定する管理者もいます。タイムアウトは、アイドル接続が持続するために許容される時間 (秒単位) です。クライアントが期間内にリクエストを送信しない場合、Redis エンジン、接続がタイムアウト値に達するとすぐに接続を終了します。タイムアウト値が小さいと、不要な切断が発生する場合があります、クライアントはそれらを適切に処理して再接続する必要があり、遅延が発生します。
- キーの格納に使用されるメモリは、クライアントバッファと共有されます。大きなリクエストまたは応答があるスロークライアントは、バッファを処理するために多くの量のメモリを要求する場合があります。Redis ElastiCache 構成のデフォルトでは、通常のクライアント出力バッファのサイズは制限されません。maxmemory の制限にヒットした場合、エンジンはバッファの使用量を満たすために項目を削除しようとしています。メモリが極端に少ない状況では、ElastiCache for Redis はメモリを解放してクラスターの状態を維持するために、大量のクライアント出力バッファを消費するクライアントを切断することを選択する場合があります。

カスタム設定を用いてクライアントバッファのサイズを制限することができ、制限をヒットしているクライアントは切断されます。ただし、クライアントは予期しない切断を処理できる必要があります。通常のクライアントのバッファサイズを処理するパラメータは次のとおりです。

- client-query-buffer-limit: 1 つの入力リクエストの最大サイズ。

- `client-output-buffer-limit-normal-soft-limit`: クライアント接続のソフトリミット。に定義されている時間 (秒) を超えてソフトリミットを超えたままの場合、`normal-soft-seconds` またはハードリミットに達した場合、接続は終了します。 `client-output-buffer-limit`
- `client-output-buffer-limit-normal-soft-seconds:-`; を超える接続の許容時間 `client-output-buffer-limit normal-soft-limit`
- `client-output-buffer-limit-normal-hard-limit`: この制限に達した接続はただちに終了します。

通常のクライアントバッファに加えて、次のオプションは、レプリカノードと Pub/Sub (Publish/Subscribe) クライアントのバッファを制御します。

- `client-output-buffer-limit-replica-hard-limit`;
- `client-output-buffer-limit-replica-soft-seconds`;
- `client-output-buffer-limit-replica-hard-limit`;
- `client-output-buffer-limit-pubsub-soft-limit`;
- `client-output-buffer-limit-pubsub-soft-seconds`;
- `client-output-buffer-limit-pubsub-hard-limit`;

Amazon EC2 インスタンスのクライアント側のトラブルシューティング

クライアント側の負荷と応答性も、へのリクエストに影響する可能性があります。ElastiCache断続的な接続性またはタイムアウトの問題のトラブルシューティングを行う際には、EC2 インスタンスおよびオペレーティングシステムの制限を慎重に確認する必要があります。観察すべきいくつかの重要なポイント：

- CPU:
 - EC2 インスタンスの CPU 使用率: CPU が飽和していない、または 100% 近くではないことを確認します。履歴分析は次の方法で行うことができますが CloudWatch、データポイントの精度は 1 分 (詳細モニタリングが有効な場合) か 5 分のどちらかであることを覚えておいてください。
 - [\[バースト可能な EC2 インスタンス\]](#) を使用する場合は、CPU クレジット残高が枯渇していないことを確認してください。CPUCreditBalance CloudWatch この情報はメトリクスで確認できません。
 - CPU 使用率が短時間高くなると、100% の使用率に反映されないままタイムアウトになる可能性があります。CloudWatchこのような場合は、`top`、`ps` および `mpstat` のようなオペレーティングシステムツールによるリアルタイムの監視が必要です。
- ネットワーク

- インスタンスの機能に応じて、ネットワークスループットが許容可能な値未満であるかどうかを確認してください。詳細については、「[Amazon EC2 のインスタンスタイプ](#)」を参照してください。
- ena 拡張ネットワークドライバーのインスタンスで、タイムアウトまたは超えられた制限がないか [\[ENA 統計\]](#) を確認してください。次の統計情報は、ネットワーク制限の飽和状態を確認するのに役立ちます。
 - bw_in_allowance_exceeded/bw_out_allowance_exceeded: 過剰なインバウンドまたはアウトバウンドのスループットのためにシェーピングされたパケット数;
 - conntrack_allowance_exceeded: セキュリティグループの [\[接続追跡制限\]](#) のためにドロップされたパケット数。この制限が飽和すると、新しい接続は失敗します。
 - linklocal_allowance_exceeded: インスタンスメタデータ、VPC DNS 経由の NTP への過剰なリクエストによりドロップされたパケット数。制限は、すべてのサービスで毎秒 1024 パケットです。
 - pps_allowance_exceeded: 1 秒あたりの過剰なパケット比率のためにドロップされたパケット数。PPS 制限は、ネットワークトラフィックが 1 秒あたり数千または数百万の非常に小さなリクエストで構成されている場合にヒットする可能性があります。ElastiCache の代わりに複数の操作を一度に実行するパイプラインやコマンドを使用して、トラフィックを最適化してネットワークパケットをより有効に活用できます。MGET GET

1 つのリクエストを完了するのにかかった時間の解読

- ネットワーク上:tcpdump と Wireshark (コマンドラインでは tshark) は、リクエストがネットワークを通過し、ElastiCache エンジンにヒットしてリターンを得るまでにかかった時間を把握するのに便利なツールです。次の例では、次のコマンドで作成された 1 つのリクエストを強調表示します。

```
$ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
+PONG
```

上記のコマンドと並行して、tcpdump が実行中であり、次のように返されました。

```
$ sudo tcpdump -i any -nn port 6379 -tt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
1609428918.917869 IP 172.31.11.142.40966
```

```
> 172.31.11.247.6379: Flags [S], seq 177032944, win 26883, options [mss
8961,sackOK,TS val 27819440 ecr 0,nop,wscale 7], length 0
1609428918.918071 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [S.], seq
53962565, ack 177032945, win
28960, options [mss 1460,sackOK,TS val 3788576332 ecr 27819440,nop,wscale 7],
length 0
1609428918.918091 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 1, win
211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918122
IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [P.], seq 1:6, ack 1, win 211,
options [nop,nop,TS val 27819440 ecr 3788576332], length 5: RESP "ping"
1609428918.918132 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [F.], seq 6, ack
1, win 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918240 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [.], ack 6, win
227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918295
IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [P.], seq 1:8, ack 7, win 227,
options [nop,nop,TS val 3788576332 ecr 27819440], length 7: RESP "PONG"
1609428918.918300 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 8, win
211, options [nop,nop,TS val 27819441 ecr 3788576332], length 0
1609428918.918302 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [F.], seq 8, ack
7, win 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918307
IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 9, win 211, options
[nop,nop,TS val 27819441 ecr 3788576332], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

上記の出力から、TCP スリーウェイハンドシェイクが 222 マイクロ秒 (918091~917869) で完了し、ping コマンドが送信され、173 マイクロ秒 (918295~918122) で返されたことを確認できます。

リクエストから接続を閉じるまで、438 マイクロ秒 (918307~917869) かかりました。これらの結果では、ネットワークとエンジンの応答時間が良好であることを確認し、調査は他のコンポーネントに焦点を当てることができます。

- オペレーティングシステム上: Strace は、OS レベルでのタイムギャップを特定するのに役立ちます。実際のアプリケーションの分析では、より広範で特殊なアプリケーションプロファイラやデバッガを使用することをお勧めします。次の例は、基本オペレーティングシステムコンポーネント

が予期したとおりに動作しているかどうかを示しています。そうでない場合、さらに調査が必要になることがあります。得られた strace を持つ同じ Redis PING コマンドを使用する。

```
$ echo ping | strace -f -tttt -r -e trace=execve,socket,open,recvfrom,sendto
nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com (http://
example.xxxxxx.ng.0001.use1.cache.amazonaws.com/)
 6379
1609430221.697712 (+ 0.000000) execve("/usr/bin/nc", ["nc",
"example.xxxxxx.ng.0001.use"..., "6379"], 0x7ffffede7cc38 /* 22 vars */) = 0
1609430221.708955 (+ 0.011231) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|
SOCK_NONBLOCK, 0) = 3
1609430221.709084
(+ 0.000124) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|SOCK_NONBLOCK, 0) = 3
1609430221.709258 (+ 0.000173) open("/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709637 (+ 0.000378) open("/etc/host.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709923
(+ 0.000286) open("/etc/resolv.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.711365 (+ 0.001443) open("/etc/hosts", O_RDONLY|O_CLOEXEC) = 3
1609430221.713293 (+ 0.001928) socket(AF_INET, SOCK_DGRAM|SOCK_CLOEXEC|SOCK_NONBLOCK,
IPPROTO_IP) = 3
1609430221.717419
(+ 0.004126) recvfrom(3, "\362|
\201\200\0\1\0\2\0\0\0\0\rnotls20201224\6tihew"..., 2048, 0, {sa_family=AF_INET,
sin_port=htons(53), sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 155
1609430221.717890 (+ 0.000469) recvfrom(3,
"\204\207\201\200\0\1\0\1\0\0\0\0\rnotls20201224\6tihew"...,
65536, 0, {sa_family=AF_INET, sin_port=htons(53),
sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 139
1609430221.745659 (+ 0.027772) socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) = 3
1609430221.747548 (+ 0.001887) recvfrom(0, 0x7ffcf2f2ca50, 8192,
0, 0x7ffcf2f2c9d0, [128]) = -1 ENOTSOCK (Socket operation on non-socket)
1609430221.747858 (+ 0.000308) sendto(3, "ping\n", 5, 0, NULL, 0) = 5
1609430221.748048 (+ 0.000188) recvfrom(0, 0x7ffcf2f2ca50, 8192, 0, 0x7ffcf2f2c9d0,
[128]) = -1 ENOTSOCK
(Socket operation on non-socket)
1609430221.748330 (+ 0.000282) recvfrom(3, "+PONG\r\n", 8192, 0, 0x7ffcf2f2c9d0,
[128->0]) = 7
+PONG
1609430221.748543 (+ 0.000213) recvfrom(3, "", 8192, 0, 0x7ffcf2f2c9d0, [128->0]) = 0
1609430221.752110
(+ 0.003569) +++ exited with 0 +++
```

上記の例では、コマンドは完了に 54 ミリ秒を若干超える時間がかかりました (752110 - 697712 = 54398 マイクロ秒)。

nc のインスタンス化と名前解決 (697712 から 717890 まで) には多くの時間 (約 20ms) がかかりました。その後、TCP ソケットの作成には 2ms (745659 から 747858)、リクエストに対する応答の送信と受信には 0.4ms (747858 から 748330) が必要でした。

Amazon のセキュリティ ElastiCache

AWS クラウドセキュリティは最優先事項です。AWS 顧客は、最もセキュリティに敏感な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の恩恵を受けることができます。

セキュリティは、AWS お客様とお客様との間で共有される責任です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- **クラウドのセキュリティ** — AWS AWS AWS クラウド内でサービスを実行するインフラストラクチャを保護する責任があります。AWS また、安全に使用できるサービスも提供します。[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 に適用されるコンプライアンスプログラムについては ElastiCache、[「AWS コンプライアンスプログラム別の対象サービス」](#)を参照してください。
- **クラウドにおけるセキュリティ** — お客様の責任は、AWS 使用するサービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon を使用する際に責任分担モデルを適用する方法を理解するのに役立ちます ElastiCache。以下のトピックでは、ElastiCache セキュリティとコンプライアンスの目標を達成するように Amazon を設定する方法を示しています。また、Amazon AWS ElastiCache リソースの監視と保護に役立つ他のサービスの使用方法についても学びます。

トピック

- [Amazon ElastiCache でのデータ保護](#)
- [インターネットトラフィックのプライバシー](#)
- [Amazon の Identity and Access Management ElastiCache](#)
- [Amazon のコンプライアンス検証 ElastiCache](#)
- [Amazon ElastiCache s の耐障害性](#)
- [AWS ElastiCache でのインフラストラクチャセキュリティ](#)
- [でのサービスの更新 ElastiCache](#)
- [一般的な脆弱性とリスク \(CVE\): Redis で対処されたセキュリティの脆弱性 ElastiCache](#)

Amazon ElastiCache でのデータ保護

AWS [責任共有モデル](#)は、AWS ElastiCache (ElastiCache) のデータ保護に適用されます。このモデルで説明したように、AWS は、すべての AWS クラウド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を負います。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用する AWS のサービスに対するセキュリティの設定と管理タスクが含まれます。データプライバシーの詳細については、「[データプライバシーのよくある質問](#)」を参照してください。

データ保護の目的で、AWS アカウントの認証情報を保護し、個々のアカウントを AWS Identity and Access Management (IAM) で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許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- TLS を使用して AWS リソースと通信します。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS 暗号化ソリューションを AWS のサービス内のすべてのデフォルトのセキュリティ管理と一緒に使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これにより、Amazon S3 に保存される個人データの検出と保護が支援されます。

顧客のアカウント番号などの機密の識別情報は、[名前] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK によって ElastiCache や他の AWS のサービスを使用する場合も同様です。ElastiCache や他のサービスに入力したすべてのデータは、診断ログに取り込まれる可能性があります。外部サーバーへの URL を指定するときは、そのサーバーへのリクエストを検証するための認証情報を URL に含めないでください。

トピック

- [Amazon ElastiCache のデータセキュリティ](#)

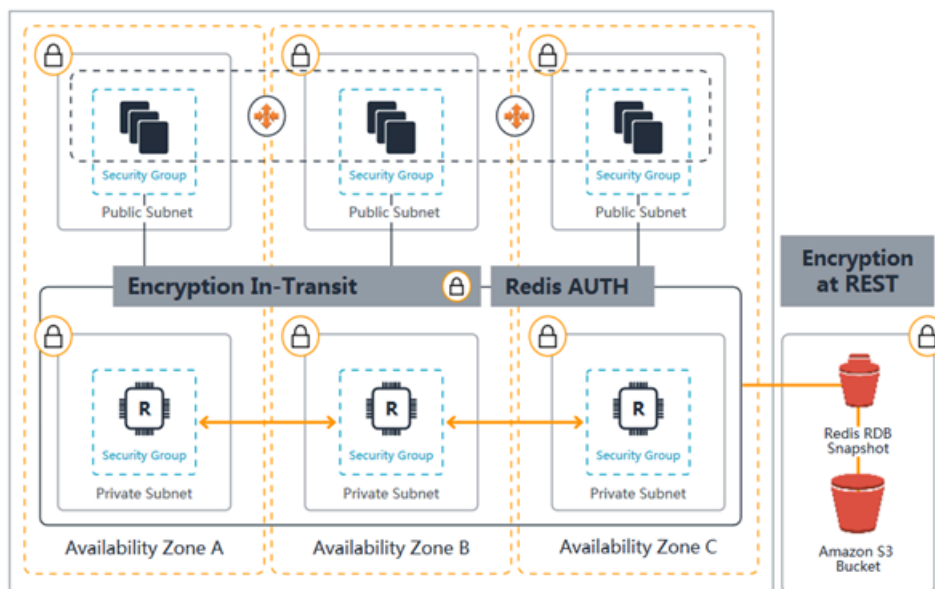
Amazon ElastiCache のデータセキュリティ

データを安全に保つために、Amazon ElastiCache および Amazon EC2 は、サーバーのデータへの不正アクセスに対する防御メカニズムを提供します。

Amazon ElastiCache for Redis は、Redis バージョン 3.2.6 (EOL の予定、「[Redis バージョンのサポート終了スケジュール](#)」を参照)、4.0.10 以降を実行しているキャッシュ上のデータに対するオプションの暗号化機能を提供します。

- 転送時の暗号化では、ある場所から別の場所へ移動するデータ (クラスターのノード間、キャッシュとアプリケーション間など) に対して暗号化が行われます。
- 保管時の暗号化では、同期やバックアップオペレーションの実行中にオンディスクデータが暗号化されます。

Amazon ElastiCache for Redis では、IAM または Redis AUTH によるユーザーの認証と、ロールベースのアクセスコントロール (RBAC) の使用によるユーザーオペレーションの承認もサポートします。



ElastiCache for Redis セキュリティの図

トピック

- [ElastiCache 転送時の暗号化 \(TLS\)](#)
- [ElastiCache での保管時の暗号化](#)
- [認証と認可](#)

ElastiCache 転送時の暗号化 (TLS)

データの安全性を維持するために、Amazon ElastiCache と Amazon EC2 はサーバー上のデータの不正アクセスから保護するメカニズムを提供します。転送時の暗号化機能 ElastiCache を提供すること

で、は、ある場所から別の場所に移動する際にデータを保護するために使用できるツールを提供します。

すべてのサーバーレスキャッシュで、転送時の暗号化が有効になっています。独自設計型クラスターでは、転送時の暗号化は、レプリケーショングループの作成時にレプリケーショングループ上でパラメータ `TransitEncryptionEnabled` を `true` (CLI: `--transit-encryption-enabled`) に設定することで有効にできます。これは、AWS Management Console、AWS CLI または ElastiCache API を使用してレプリケーショングループを作成する場合でも実行できます。

トピック

- [転送時の暗号化の概要](#)
- [転送時の暗号化の条件](#)
- [転送時の暗号化のベストプラクティス](#)
- [以下も参照してください。](#)
- [転送時の暗号化を有効にする](#)
- [redis-cli を使用して転送時の暗号化で Amazon ElastiCache for Redis に接続する](#)
- [独自設計型 Redis クラスターで Python を使用して転送時の暗号化を有効にする](#)
- [転送時の暗号化を有効にする際のベストプラクティス](#)

転送時の暗号化の概要

Amazon ElastiCache 転送時の暗号化は、ある場所から別の場所への転送中に、最も脆弱なポイントでデータのセキュリティを強化できる機能です。エンドポイントでデータの暗号化と復号を行うにはある程度の処理が必要であるため、転送時の暗号化を有効にするとパフォーマンスに影響を及ぼす可能性があります。転送時の暗号化の使用時と未使用時でデータのベンチマークを取得して、ユースケースにおけるパフォーマンス影響を判断する必要があります。

ElastiCache 転送時の暗号化には、次の機能が実装されています。

- 暗号化されたクライアント接続 — キャッシュノードへのクライアント接続は TLS で暗号化されます。
- 暗号化されたサーバー接続 — クラスター内のノード間を移動するデータは暗号化されます。
- [サーバー認証] — クライアントは、適切なサーバーに接続していることを認証できます。
- [クライアント認証] — Redis AUTH 機能を使用して、サーバーはクライアントを認証できます。

転送時の暗号化の条件

独自設計型クラスターの実装を計画する際には、Amazon の転送 ElastiCache 時の暗号化に関する以下の制約事項に留意する必要があります。

- 転送時の暗号化は、Redis バージョン 3.2.6 および 4.0.10 以降を実行するレプリケーショングループでサポートされます。
- 既存のクラスターにおける転送中の暗号化設定の変更は、Redis バージョン 7 以降を実行しているレプリケーショングループでサポートされています。
- 転送時の暗号化は、Amazon VPC. で実行しているレプリケーショングループでのみサポートされます。
- 転送時の暗号化は、MM1, M2 のノードタイプを実行するレプリケーショングループではサポートされていません。

詳細については、「[サポートされているノードの種類](#)」を参照してください。

- 転送時の暗号化は、パラメータ `TransitEncryptionEnabled` を `true` に明示的に設定することで有効化されます。
- キャッシュクライアントが TLS 接続をサポートしていることと、クライアント設定で TLS 接続を有効にしていることを確認します。
- 古い TLS 1.0 および TLS 1.1 の使用は、ElastiCache バージョン 6 以降ではすべての AWS リージョンで廃止されました。ElastiCache は、2025 年 5 月 8 日まで TLS 1.0 および 1.1 を引き続きサポートします。お客様は、その日付より前にクライアントソフトウェアを更新する必要があります。

転送時の暗号化のベストプラクティス

- エンドポイントでデータの暗号化と復号を行うにはある程度の処理が必要であるため、転送時の暗号化の実装によりパフォーマンスが低下する可能性があります。自身のデータで転送時の暗号化使用時のベンチマークを暗号化なしの場合と比較して、実装におけるパフォーマンスの影響を判断してください。
- 新しい接続の作成には高い負荷がかかる場合があるため、TLS 接続を持続させることで転送時の暗号化のパフォーマンスへの影響を軽減させることができます。

以下も参照してください。

- [ElastiCache での保管時の暗号化](#)

- [Redis AUTH コマンドによる認証](#)
- [ロールベースのアクセスコントロール \(RBAC\) を使用したユーザーの認証](#)
- [Amazon VPC と ElastiCache のセキュリティ](#)
- [Amazon の Identity and Access Management ElastiCache](#)

転送時の暗号化を有効にする

すべてのサーバーレスキャッシュで、転送時の暗号化が有効になっています。独自設計型クラスターで、AWS Management Console、AWS CLI、または ElastiCache API を使用して転送時の暗号化を有効にできます。

AWS Management Console を使用して転送時の暗号化を有効にする

AWS Management Console を使用して新しい独自設計型クラスターで転送時の暗号化を有効にする

独自のクラスターを設計する場合、「簡単な作成」方式の「開発/テスト」構成と「本番稼働用」構成では、転送時の暗号化が有効になっています。設定を自分で選択するときは、以下のように選択します。

- エンジンバージョン 3.2.6 または 4.0.10 以降。
- [転送中の暗号化] オプションの [有効化] の横にあるチェックボックスをオンにします。

詳しいステップについては、以下を参照ください。

- [Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)

AWS Management Console を使用して既存の独自設計型クラスターで転送時の暗号化を有効にする

転送中の暗号化を有効にするには、2段階のプロセスが必要です。まず、転送中の暗号化モードを preferred に設定する必要があります。このモードでは、Redis クライアントは暗号化された接続と暗号化されていない接続の両方を使用して接続できます。暗号化接続を使用するようにすべての Redis クライアントを移行したら、クラスター設定を変更して転送中の暗号化モードを required に設定できます。転送中の暗号化モードを required に設定すると、暗号化されていない接続はすべてドロップされ、暗号化された接続のみが許可されます。

ステップ 1: [Transit encryption mode] (転送中の暗号化モード) を [Preferred] (優先) に設定する

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 左側にあるナビゲーションペインに表示されている ElastiCache の [リソース] から [Redis キャッシュ] を選択します。
3. 更新する [Redis キャッシュ] を選択します。
4. [Actions] (アクション) ドロップダウンを選択してから、[Modify] (変更) を選択します。
5. [Security] (セキュリティ) セクションの [Encryption in transit] (転送時の暗号化) で [Enable] (有効化) を選択します。
6. [Transit encryption mode] (転送中の暗号化モード) として [Preferred] (優先) を選択します。
7. 変更を行ってから、[Preview changes] (変更のプレビュー) を選択します。

すべての Redis クライアントを暗号化接続の使用に移行した後:

ステップ 2: [Transit encryption mode] (転送中の暗号化モード) を [Required] (必須) に設定する

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 左側にあるナビゲーションペインに表示されている ElastiCache の [リソース] から [Redis キャッシュ] を選択します。
3. 更新する [Redis キャッシュ] を選択します。
4. [Actions] (アクション) ドロップダウンを選択してから、[Modify] (変更) を選択します。
5. [Security] (セキュリティ) セクションの [Transit encryption mode] (転送中の暗号化モード) として [Required] (必須) を選択します。
6. 変更を行ってから、[Preview changes] (変更のプレビュー) を選択します。

AWS CLI を使用して転送時の暗号化を有効にする

AWS CLI を使用して Redis レプリケーショングループの作成時に転送時の暗号化を有効にするには、パラメータ `transit-encryption-enabled` を使用します。

Redis の新しい独自設計型クラスター (クラスターモードが無効) 上で転送時の暗号化を有効にする (CLI)

AWS CLI オペレーション `create-replication-group` と以下のパラメータを使用して、転送時の暗号化が有効な、レプリカを持つ Redis レプリケーショングループを作成します。

主要パラメータ:

- **--engine** — `redis` を指定する必要があります。
- **--engine-version** — 3.2.6 または 4.0.10 以降を指定する必要があります。
- **--transit-encryption-enabled** — 必須。転送時の暗号化を有効にする場合、`--cache-subnet-group` パラメータの値も指定する必要があります。
- **--num-cache-clusters** — 1 以上を指定する必要があります。このパラメータの最大値は 6 です。

詳細については、次を参照してください:

- [Redis \(クラスターモードが無効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)
- [create-replication-group](#)

Redis の新しい独自設計型クラスター (クラスターモードが有効) 上で転送時の暗号化を有効にする (CLI)

AWS CLI オペレーション `create-replication-group` と以下のパラメータを使用して、転送時の暗号化が有効な Redis (クラスターモードが有効) レプリケーショングループを作成します。

主要パラメータ:

- **--engine** — `redis` を指定する必要があります。
- **--engine-version** — 3.2.6 または 4.0.10 以降を指定する必要があります。
- **--transit-encryption-enabled** — 必須。転送時の暗号化を有効にする場合、`--cache-subnet-group` パラメータの値も指定する必要があります。
- 次のいずれかのパラメータセットを使用して、レプリケーショングループのノードグループの構成を指定します。
 - **--num-node-groups** — このレプリケーショングループ内のシャード数 (ノードグループ数) を指定します。このパラメータの最大値は 500 です。

--replicas-per-node-group — 各ノードグループ (シャード) のレプリカノードの数を指定します。ここで指定される値は、このレプリケーショングループのすべてのシャードに適用されます。このパラメータの最大値は 5 です。

- **--node-group-configuration** — 各シャードの構成を個別に指定します。

詳細については、次を参照してください:

- [Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)
- [create-replication-group](#)

AWS CLI を使用して既存のクラスターで転送時の暗号化を有効にする

転送中の暗号化を有効にするには、2 段階のプロセスが必要です。まず、転送中の暗号化モードを `preferred` に設定する必要があります。このモードでは、Redis クライアントは暗号化された接続と暗号化されていない接続の両方を使用して接続できます。暗号化接続を使用するようにすべての Redis クライアントを移行したら、クラスター設定を変更して転送中の暗号化モードを `required` に設定できます。転送中の暗号化モードを `required` に設定すると、暗号化されていない接続はすべてドロップされ、暗号化された接続のみが許可されます。

AWS CLI オペレーション `modify-replication-group` と以下のパラメータを使用して、転送時の暗号化が無効な Redis (クラスターモードが有効) レプリケーショングループを更新します。

転送中の暗号化を有効にするには

1. 次のパラメータを使用して、転送暗号化モードを `preferred` に設定する

- **--transit-encryption-enabled** — 必須。
- **--transit-encryption-mode** — `preferred` に設定する必要があります。

2. 次のパラメータを使用して、転送暗号化モードを `required` に設定します。

- **--transit-encryption-enabled** — 必須。
- **--transit-encryption-mode** — `required` に設定する必要があります。

`redis-cli` を使用して転送時の暗号化で Amazon ElastiCache for Redis に接続する

転送時の暗号化が有効になっている Redis キャッシュ ElastiCache の からデータにアクセスするには、Secure Socket Layer (SSL) を使用するクライアントを使用します。Amazon Linux や Amazon

Linux 2 で、TLS/SSL を使用して redis-cli を使用することもできます。クライアントが TLS に対応していない場合は、クライアントホストで stunnel コマンドを使用して、Redis ノードへの SSL トンネルを作成できます。

Linux との暗号化された接続

redis-cli を使用して、Amazon Linux 2023、Amazon Linux 2、または Amazon Linux で転送時の暗号化が有効になっている Redis クラスターに接続するには、次の手順に従います。

1. redis-cli ユーティリティをダウンロードし、コンパイルします。このユーティリティは Redis ソフトウェアディストリビューションに含まれています。
2. EC2 インスタンスのコマンドプロンプトで、使用している Linux のバージョンに適したコマンドを入力します。

Amazon Linux 2023

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

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

次に、次のコマンドを入力し、クラスターのエンドポイントとポートをこの例に示されているものに置き換えます。

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

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

Amazon Linux 2

Amazon Linux 2 を使用している場合は、次のように入力します。

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

Amazon Linux

Amazon Linux を使用している場合は、次のように入力します。

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

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

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

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

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

Note

Amazon Linux 2023 に redis6 をインストールする場合、redis6-cliの代わりにコマンドを使用できるようになりましたredis-cli。

```
redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
```

stunnel を使用した暗号化された接続

redis-cli を使用して、stunnel を使用した転送時の暗号化が有効になっている Redis クラスターに接続するには、次の手順に従います。

1. SSH を使用してクライアントに接続し、stunnel をインストールします。

```
sudo yum install stunnel
```

2. 次のコマンドを実行して、ファイル '/etc/stunnel/redis-cli.conf' を同時に作成および編集し、以下の出力をテンプレートとして使用して、1 つ以上の接続パラメータに ElastiCache for Redis クラスターエンドポイントを追加します。

```
vi /etc/stunnel/redis-cli.conf

fips = no
setuid = root
setgid = root
pid = /var/run/stunnel.pid
debug = 7
delay = yes
options = NO_SSLv2
options = NO_SSLv3
[redis-cli]
  client = yes
  accept = 127.0.0.1:6379
  connect = primary.ssltest.wif01h.use1.cache.amazonaws.com:6379
[redis-cli-replica]
  client = yes
  accept = 127.0.0.1:6380
  connect = ssltest-02.ssltest.wif01h.use1.cache.amazonaws.com:6379
```

この例では、設定ファイルに `redis-cli` と `redis-cli-replica` という 2 つの接続があります。パラメータは次のように設定されます。

- この `stunnel` インスタンスがクライアントであることを指定するために、`client` は `yes` に設定されています。
- `accept` はクライアント IP に設定されています。この例では、プライマリはポート 6379 の Redis のデフォルト 127.0.0.1 に設定されています。レプリカは別のポートを呼び出し、6380 に設定する必要があります。エフェメラルポート 1024 ~ 65535 を使用できます。詳細については、Amazon VPC ユーザーガイドの「[一時ポート](#)」を参照してください。
- `connect` は Redis サーバーエンドポイントに設定されています。詳細については、「[接続エンドポイントの検索](#)」を参照してください。

3. `stunnel` を起動します。


```
sudo stunnel /etc/stunnel/redis-cli.conf
```

netstat コマンドを使用して、トンネルが開始されたことを確認します。

```
sudo netstat -tulnp | grep -i stunnel
```

```
tcp        0      0 127.0.0.1:6379          0.0.0.0:*        LISTEN
           3189/stunnel
tcp        0      0 127.0.0.1:6380          0.0.0.0:*        LISTEN
           3189/stunnel
```

4. トンネルのローカルエンドポイントを使用して、暗号化された Redis ノードに接続します。
 - ElastiCache Redis クラスターの作成中に AUTH パスワードが使用されなかった場合、この例では redis-cli を使用して、Amazon Linux で redis-cli ElastiCache の完全パスを使用して for Redis サーバーに接続します。

```
/home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379
```

Redis クラスターの作成中に AUTH パスワードが使用された場合、この例では redis-cli を使用して、Amazon Linux で redis-cli の完全パスを使用して Redis サーバーに接続します。

```
/home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379 -a my-secret-password
```

または

- ディレクトリを redis-stable に変更し、次の操作を行います。

ElastiCache Redis クラスターの作成中に AUTH パスワードが使用されなかった場合、この例では redis-cli を使用して、Amazon Linux で redis-cli ElastiCache の完全パスを使用して for Redis サーバーに接続します。

```
src/redis-cli -h localhost -p 6379
```

Redis クラスターの作成中に AUTH パスワードが使用された場合、この例では redis-cli を使用して、Amazon Linux で redis-cli の完全パスを使用して Redis サーバーに接続します。

```
src/redis-cli -h localhost -p 6379 -a my-secret-password
```

この例では、Telnet を使用して Redis サーバーに接続します。

```
telnet localhost 6379

Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
auth MySecretPassword
+OK
get foo
$3
bar
```

5. SSL トンネルを停止して閉じるには、`pkill stunnel` プロセスを実行します。

```
sudo pkill stunnel
```

独自設計型 Redis クラスターで Python を使用して転送時の暗号化を有効にする

次のガイドでは、転送中の暗号化を無効にして最初に作成された Redis 7.0 クラスターで転送中の暗号化を有効にする方法を説明します。TCP クライアントと TLS クライアントは、このプロセス中もダウンタイムなしでクラスターとの通信を継続します。

Boto3 は環境変数から必要な認証情報 (`aws_access_key_id`、`aws_secret_access_key`、および `aws_session_token`) を取得します。これらの認証情報は、このガイドに示されている Python コードを処理するために `python3` を実行するのと同じ `bash` ターミナルにあらかじめ貼り付けておきます。以下の例のコードは、同じ VPC で起動された EC2 インスタンスからのプロセスであり、その中に ElastiCache Redis クラスターを作成するために使用されます。

Note

- 次の例では、ElastiCache 管理オペレーション (クラスターまたはユーザーの作成) で boto3 SDK を使用し、データ処理で `redis-py/redis-py-cluster` を使用します。
- クラスター変更 API でオンライン TLS 移行を使用するには、boto3 バージョン (=~) 1.26.39 以上を使用する必要があります。

- ElastiCache は、バージョン 7.0 以降の Redis クラスターに対してのみオンライン TLS 移行をサポートします。そのため、7.0 より前のバージョンの Redis を実行しているクラスターがある場合は、クラスターの Redis バージョンをアップグレードする必要があります。バージョンの違いの詳細については、「[メジャーバージョンの動作と互換性の違い](#)」を参照してください。

トピック

- [ElastiCache Redis クラスターを起動する文字列定数を定義する](#)
- [クラスター構成用のクラスを定義する](#)
- [クラスター自体を表すクラスを定義する](#)
- [\(オプション\) Redis クラスターへのクライアント接続をデモするためのラッパークラスの作成](#)
- [転送中の暗号化設定を変更するプロセスをデモする main 関数を作成する](#)

ElastiCache Redis クラスターを起動する文字列定数を定義する

まず security-group、Cache Subnet group、default parameter group など、ElastiCache クラスターの作成に必要な AWS エンティティの名前を保持するシンプルな Python 文字列定数を定義しましょう。これらの AWS エンティティはすべて、使用するリージョンの AWS アカウントで事前に作成する必要があります。

```
#Constants definitions
SECURITY_GROUP = "sg-0492aa0a29c558427"
CLUSTER_DESCRIPTION = "This cluster has been launched as part of the online TLS
migration user guide"
EC_SUBNET_GROUP = "client-testing"
DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED = "default.redis7.cluster.on"
```

クラスター構成用のクラスを定義する

ここで、クラスターの設定を表すシンプルな Python クラスをいくつか定義しましょう。このクラスには、Redis のバージョン、インスタンスタイプ、転送中の暗号化 (TLS) が有効か無効かなど、クラスターに関するメタデータが格納されます。

```
#Class definitions

class Config:
    def __init__(
```

```
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
    ):
        self.instance_type = instance_type
        self.version = version
        self.multi_az = multi_az
        self.TLS = TLS
        self.name = name or f"tls-test"

    def create_base_launch_request(self):
        return {
            "ReplicationGroupId": self.name,
            "TransitEncryptionEnabled": self.TLS,
            "MultiAZEnabled": self.multi_az,
            "CacheNodeType": self.instance_type,
            "Engine": "redis",
            "EngineVersion": self.version,
            "CacheSubnetGroupName": EC_SUBNET_GROUP ,
            "CacheParameterGroupName":
DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED ,
            "ReplicationGroupDescription": CLUSTER_DESCRIPTION,
            "SecurityGroupIds": [SECURITY_GROUP],
        }

class ConfigCME(Config):
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
        num_shards: int = 2,
        num_replicas_per_shard: int = 1,
    ):
        super().__init__(instance_type, version, multi_az, TLS, name)
        self.num_shards = num_shards
        self.num_replicas_per_shard = num_replicas_per_shard

    def create_launch_request(self) -> dict:
```

```
launch_request = self.create_base_launch_request()
launch_request["NumNodeGroups"] = self.num_shards
launch_request["ReplicasPerNodeGroup"] = self.num_replicas_per_shard
return launch_request
```

クラスター自体を表すクラスを定義する

次に、ElastiCache Redis クラスター自体を表すシンプルな Python クラスをいくつか定義しましょう。このクラスには、クラスターの作成や ElastiCache API のクエリなどの ElastiCache 管理オペレーションに使用する boto3 クライアントを格納するクライアントフィールドがあります。

```
import botocore.config
import boto3

# Create boto3 client
def init_client(region: str = "us-east-1"):
    config = botocore.config.Config(retries={"max_attempts": 10, "mode": "standard"})
    init_request = dict()
    init_request["config"] = config
    init_request["service_name"] = "elasticache"
    init_request["region_name"] = region
    return boto3.client(**init_request)

class ElastiCacheClusterBase:
    def __init__(self, name: str):
        self.name = name
        self.elasticache_client = init_client()

    def get_first_replication_group(self):
        return self.elasticache_client.describe_replication_groups(
            ReplicationGroupId=self.name
        )["ReplicationGroups"][0]

    def get_status(self) -> str:
        return self.get_first_replication_group()["Status"]

    def get_transit_encryption_enabled(self) -> bool:
        return self.get_first_replication_group()["TransitEncryptionEnabled"]

    def is_available(self) -> bool:
        return self.get_status() == "available"
```

```
def is_modifying(self) -> bool:
    return self.get_status() == "modifying"

def wait_for_available(self):
    while True:
        if self.is_available():
            break
        else:
            time.sleep(5)

def wait_for_modifying(self):
    while True:
        if self.is_modifying():
            break
        else:
            time.sleep(5)

def delete_cluster(self) -> bool:
    self.elasticache_client.delete_replication_group(
        ReplicationGroupId=self.name, RetainPrimaryCluster=False
    )

def modify_transit_encryption_mode(self, new_transit_encryption_mode: str):
    # generate api call to migrate the cluster to TLS preferred or to TLS required
    self.elasticache_client.modify_replication_group(
        ReplicationGroupId=self.name,
        TransitEncryptionMode=new_transit_encryption_mode,
        TransitEncryptionEnabled=True,
        ApplyImmediately=True,
    )
    self.wait_for_modifying()

class ElastiCacheClusterCME(ElastiCacheClusterBase):
    def __init__(self, name: str):
        super().__init__(name)

    @classmethod
    def launch(cls, config: ConfigCME = None) -> ElastiCacheClusterCME:
        config = config or ConfigCME()
        print(config)
        new_cluster = ElastiCacheClusterCME(config.name)
        launch_request = config.create_launch_request()
        new_cluster.elasticache_client.create_replication_group(**launch_request)
        new_cluster.wait_for_available()
```

```
        return new_cluster

    def get_configuration_endpoint(self) -> str:
        return self.get_first_replication_group()["ConfigurationEndpoint"]["Address"]

#Since the code can throw exceptions, we define this class to make the code more
readable and
#so we won't forget to delete the cluster
class ElastiCacheCMEManager:
    def __init__(self, config: ConfigCME = None):
        self.config = config or ConfigCME()

    def __enter__(self) -> ElastiCacheClusterCME:
        self.cluster = ElastiCacheClusterCME.launch(self.config)
        return self.cluster

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cluster.delete_cluster()
```

(オプション) Redis クラスターへのクライアント接続をデモするためのラッパークラスの作成

次に、redis-py-cluster クライアント用のラッパークラスを作成しましょう。このラッパークラスは、クラスターにいくつかのキーをあらかじめ入力してから、ランダムに繰り返し get コマンドを実行することをサポートします。

Note

これはオプションのステップですが、後のステップに含まれる main 関数のコードが簡略化されます。

```
import redis
import random
from time import perf_counter_ns, time

class DowntimeTestClient:
    def __init__(self, client):
        self.client = client

    # num of keys prefilled
    self.prefilled = 0
```

```
# percent of get above prefilled
self.percent_get_above_prefilled = 10 # nil result expected when get hit above
prefilled
# total downtime in nano seconds
self.downtime_ns = 0
# num of success and fail operations
self.success_ops = 0
self.fail_ops = 0
self.connection_errors = 0
self.timeout_errors = 0

def replace_client(self, client):
    self.client = client

def prefill_data(self, timelimit_sec=60):
    end_time = time() + timelimit_sec
    while time() < end_time:
        self.client.set(self.prefilled, self.prefilled)
        self.prefilled += 1

# unsuccessful operations throw exceptions
def _exec(self, func):
    try:
        start_ns = perf_counter_ns()
        func()
        self.success_ops += 1
        elapsed_ms = (perf_counter_ns() - start_ns) // 10 ** 6
        # upon succesful execution of func
        # reset random_key to None so that the next command
        # will use a new random key
        self.random_key = None

    except Exception as e:
        elapsed_ns = perf_counter_ns() - start_ns
        self.downtime_ns += elapsed_ns
        # in case of failure- increment the relevant counters so that we will keep
track
        # of how many connection issues we had while trying to communicate with
        # the cluster.
        self.fail_ops += 1
        if e.__class__ is redis.exceptions.ConnectionError:
            self.connection_errors += 1
        if e.__class__ is redis.exceptions.TimeoutError:
```



```
        self.timeout_errors += 1

def _repeat_exec(self, func, seconds):
    end_time = time() + seconds
    while time() < end_time:
        self._exec(func)

def _new_random_key_if_needed(self, percent_above_prefilled):
    if self.random_key is None:
        max = int((self.prefilled * (100 + percent_above_prefilled)) / 100)
        return random.randint(0, max)
    return self.random_key

def _random_get(self):
    key = self._new_random_key_if_needed(self.percent_get_above_prefilled)
    result = self.client.get(key)
    # we know the key was set for sure only in the case key < self.prefilled
    if key < self.prefilled:
        assert result.decode("UTF-8") == str(key)

def repeat_get(self, seconds=60):
    self._repeat_exec(self._random_get, seconds)

def get_downtime_ms(self) -> int:
    return self.downtime_ns // 10 ** 6

def do_get_until(self, cond_check):
    while not cond_check():
        self.repeat_get()
    # do one more get cycle once condition is met
    self.repeat_get()
```

転送中の暗号化設定を変更するプロセスをデモする main 関数を作成する

それでは、次の処理を行う main 関数を定義しましょう。

1. boto3 ElastiCache クライアントを使用してクラスターを作成します。
2. TLS を使用しないクリア TCP 接続でクラスターに接続する redis-py-cluster クライアントを初期化します。
3. redis-py-cluster クライアントはクラスターにいくつかのデータを事前入力します。

4. boto3 クライアントは、TLS なしから TLS 優先への TLS 移行をトリガーします。
5. クラスターが TLS Preferred に移行されている間、redis-py-cluster TCP クライアントは、移行が完了するまでクラスターに繰り返し get オペレーションを送信します。
6. TLS Preferred への移行が完了したら、クラスターが転送中の暗号化をサポートしていることを確認します。その後、TLS を使用してクラスターに接続する redis-py-cluster クライアントを作成します。
7. 新しい TLS クライアントと古い TCP クライアントを使用して、いくつかの get コマンドを送信します。
8. boto3 クライアントは、TLS Preferred から TLS 必須への TLS 移行をトリガーします。
9. クラスターが TLS 必須に移行されている間、redis-py-cluster TCP クライアントは、移行が完了するまでクラスターに繰り返し get オペレーションを送信します。

```
import redis

def init_cluster_client(
    cluster: ElastiCacheClusterCME, prefill_data: bool, TLS: bool = True) ->
DowntimeTestClient:
    # we must use for the host name the cluster configuration endpoint.
    redis_client = redis.RedisCluster(
        host=cluster.get_configuration_endpoint(), ssl=TLS, socket_timeout=0.25,
        socket_connect_timeout=0.1
    )
    test_client = DowntimeTestClient(redis_client)
    if prefill_data:
        test_client.prefill_data()
    return test_client

if __name__ == '__main__':
    config = ConfigCME(TLS=False, instance_type="cache.m5.large")

    with ElastiCacheCMEManager(config) as cluster:
        # create a client that will connect to the cluster with clear tcp connection
        test_client_tcp = init_cluster_client(cluster, prefill_data=True, TLS=False)

        # migrate the cluster to TLS Preferred
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="preferred")

        # do repeated get commands until the cluster finishes the migration to TLS
        Preferred
```

```
test_client_tcp.do_get_until(cluster.is_available)

# verify that in transit encryption is enabled so that clients will be able to
connect to the cluster with TLS
assert cluster.get_transit_encryption_enabled() == True

# create a client that will connect to the cluster with TLS connection.
# we must first make sure that the cluster indeed supports TLS
test_client_tls = init_cluster_client(cluster, prefill_data=True, TLS=True)

# by doing get commands with the tcp client for 60 more seconds
# we can verify that the existing tcp connection to the cluster still works
test_client_tcp.repeat_get(seconds=60)

# do get commands with the new TLS client for 60 more seconds
test_client_tcp.repeat_get(seconds=60)

# migrate the cluster to TLS required
cluster.modify_transit_encryption_mode(new_transit_encryption_mode="required")

# from this point the tcp clients will be disconnected and we must not use them
anymore.
# do get commands with the TLS client until the cluster finishes migration to
TLS required mode.
test_client_tls.do_get_until(cluster.is_available)
```

転送時の暗号化を有効にする際のベストプラクティス

転送中の暗号化を有効にする前: DNS レコードが適切に処理されていることを確認

Note

このプロセス中に、古いエンドポイントを変更および削除します。エンドポイントの使い方を誤ると、Redis クライアントが古くて削除されたエンドポイントを使用し、クラスターに接続できなくなる可能性があります。

クラスターが TLS なしから TLS 優先に移行している間、古いノードごとの DNS レコードは保持され、新しいノードごとの DNS レコードは別の形式で生成されます。TLS が有効なクラスターは、TLS が有効でないクラスターとは異なる形式の DNS レコードを使用します。クラスターが [暗号化モード: 推奨] に設定されている場合、ElastiCache は両方の DNS レコードを保持します。これ

により、アプリケーションや他の Redis クライアントは、それらを切り替えることができます。TLS 移行プロセス中に、DNS レコードの次の変更が行われます。

転送中の暗号化を有効にした場合に行われる DNS レコードの変更の説明

CME クラスターの場合

クラスターが [転送暗号化モード: 優先] に設定されている場合:

- TLS が有効になっていないクラスターの元のクラスターエンドポイントはアクティブなままになります。クラスターを TLS 暗号化モード [なし] から [優先] に再設定しても、ダウンタイムは発生しません。
- クラスターが TLS 優先モードに設定されると、新しい TLS Redis エンドポイントが生成されます。これらの新しいエンドポイントは、古いエンドポイントと同じ IP (TLS なし) に解決されます。
- 新しい TLS Redis 設定エンドポイントは ElastiCache コンソールと describe-replication-group API へのレスポンスで公開されます。

クラスターが [転送暗号化モード: 必須] に設定されている場合:

- TLS が有効になっていない古いエンドポイントは削除されます。TLS クラスターエンドポイントのダウンタイムは発生しません。
- 新しい cluster-configuration-endpoint は、ElastiCache コンソールまたは describe-replication-group API から取得できます。

自動フェイルオーバーが有効または自動フェイルオーバーが無効な CMD クラスターの場合

レプリケーショングループが [転送暗号化モード: 優先] に設定されている場合:

- TLS が有効になっていないクラスターの元のプライマリエンドポイントとリーダーエンドポイントはアクティブなままです。
- クラスターが TLS Preferred モードに設定されると、新しいプライマリエンドポイントとリーダーエンドポイントが生成されます。この新しいエンドポイントは、古いエンドポイントと同じ IP (TLS なし) に解決されます。
- 新しいプライマリエンドポイントとリーダーエンドポイントは ElastiCache コンソールと describe-replication-group API へのレスポンスで公開されます。

レプリケーショングループが [転送暗号化モード: 必須] に設定されている場合:

- TLS が有効になっていないクラスターの元のプライマリエンドポイントとリーダーエンドポイントはアクティブなままです。
- 古い TLS なしのプライマリエンドポイントとリーダーエンドポイントは削除されます。TLS クラスターエンドポイントのダウンタイムは発生しません。
- 新しいプライマリエンドポイントとリーダーエンドポイントは、ElastiCache コンソールまたは `describe-replication-group` API から取得できます。

DNS レコードの推奨される使用方法

CME クラスターの場合

- アプリケーションのコードでは、ノードごとの DNS レコードの代わりにクラスター設定エンドポイントを使用してください。シャードの追加や削除時に変更される可能性があるため、ノードごとの DNS 名を直接使用することはお勧めしません。
- このプロセス中に変更されるため、アプリケーションでクラスター設定エンドポイントをハードコードしないでください。
- このプロセス中に変更される可能性があるため、クラスター設定エンドポイントをアプリケーションにハードコードすることは推奨されません。転送中の暗号化が完了したら、`describe-replication-group` API (上記太字で表記) を使用してクラスター設定エンドポイントをクエリし、レスポンスとして取得した DNS をこの時点から使用します。

自動フェイルオーバーが有効になっている CMD クラスターの場合

- クラスターを TLS なしから TLS 優先に移行すると、古いノードごとの DNS 名が削除され、新しい DNS 名が生成されるため、アプリケーションのコードではノードごとの DNS 名の代わりにプライマリエンドポイントとリーダーエンドポイントを使用してください。将来クラスターにレプリカを追加する可能性があるため、ノードごとの DNS 名を直接使用することはお勧めしません。また、自動フェイルオーバーが有効になっている場合、プライマリクラスターとレプリカのロールが ElastiCache サービスによって自動的に変更されます。これらの変更を追跡しやすくするために、プライマリエンドポイントとリーダーエンドポイントを使用することをお勧めします。最後に、リーダーエンドポイントを使用すると、レプリカからの読み取りをクラスター内のレプリカ間で均等に分散できます。
- TLS 移行プロセス中に変更される可能性があるため、プライマリエンドポイントとリーダーエンドポイントをアプリケーションにハードコードすることはお勧めしません。TLS 優先への移行が

完了したら、describe-replication-group API を使用してプライマリエンドポイントとリーダーエンドポイントをクエリし、レスポンスとして取得した DNS をこの時点から使用します。これにより、エンドポイントの変更を動的に追跡できます。

自動フェイルオーバーが無効になっている CMD クラスターの場合

- アプリケーションのコードでは、ノードごとの DNS 名の代わりに、プライマリエンドポイントとリーダーエンドポイントを使用します。自動フェイルオーバーが無効になっている場合、スケールリング、パッチ、フェイルオーバー、および自動フェイルオーバーが有効な場合に ElastiCache サービスによって自動的に管理されるその他の手順は、代わりにユーザーが実行します。これにより、さまざまなエンドポイントを手動で追跡することが容易になります。クラスターを TLS なしから TLS 優先に移行する際に、古いノードごとの DNS 名は削除され、新しい DNS 名が生成されるため、ノードごとの DNS 名を直接使用しないでください。これは、TLS 移行中にクライアントがクラスターに接続するために必須です。また、リーダーエンドポイントを使用する場合、レプリカ間で読み取りを均等に分散させ、クラスターにレプリカを追加または削除するときに DNS レコードを追跡できるという利点もあります。
- TLS 移行プロセス中に変更される可能性があるため、クラスター設定エンドポイントをアプリケーションにハードコードすることは推奨されません。

転送中の暗号化中: 移行プロセスが終了するタイミングに注意

転送中の暗号化モードの変更は、即座に行われるわけではなく、ある程度の時間を要することがあります。これは、大規模なクラスターの場合に特に当てはまります。クラスターが TLS 優先への移行を完了した場合にのみ、TCP 接続と TLS 接続の両方を受け入れてサービスを提供できるようになります。そのため、転送中の暗号化が完了するまでは、クラスターへの TLS 接続を確立しようとするクライアントを作成しないでください。

転送中の暗号化が成功または失敗したときに通知を受け取る方法はいくつかあります (上記のコード例には示されていません)。

- SNS サービスを使用して暗号化が完了したときに通知を受け取る
- 暗号化が完了するとイベントを発行する describe-events API を使用する
- ElastiCache コンソールに暗号化が完了したことを示すメッセージが表示される

暗号化が完了したかどうかを確認するロジックをアプリケーションに実装することもできます。上の例では、クラスターが移行を完了したことを確認する方法をいくつか見てきました。

- 移行プロセスが開始される (クラスターのステータスが「変更中」に変わる) まで待ち、変更が完了する (クラスターのステータスが「使用可能」に戻る) まで待つ
- describe-replication-group API をクエリして、クラスターの transit_encryption_enabled が [True] に設定されていることを確認する。

転送中の暗号化を有効にした後: 使用するクライアントが正しく設定されていることを確認

クラスターが TLS 優先モードになっている間、アプリケーションはクラスターへの TLS 接続を開き、それらの接続のみを使用する必要があります。これにより、転送中の暗号化を有効にしてもアプリケーションのダウンタイムは発生しません。SSL セクションの Redis の info コマンドを使用すると、Redis エンジンへのクリアな TCP 接続がないことを確認できます。

```
# SSL
ssl_enabled:yes
ssl_current_certificate_not_before_date:Mar 20 23:27:07 2017 GMT
ssl_current_certificate_not_after_date:Feb 24 23:27:07 2117 GMT
ssl_current_certificate_serial:D8C7DEA91E684163
tls_mode_connected_tcp_clients:0 (should be zero)
tls_mode_connected_tls_clients:100
```

ElastiCache での保管時の暗号化

データを安全に保つために、Amazon ElastiCache と Amazon S3 には、キャッシュ内のデータへのアクセスを制限するさまざまな方法が用意されています。詳細については、[Amazon VPC と ElastiCache のセキュリティ](#) および [Amazon の Identity and Access Management ElastiCache](#) を参照してください。

ElastiCache の保管時の暗号化は、ディスク上のデータを暗号化することでデータのセキュリティを強化する機能です。この機能はサーバーレスキャッシュでは常に有効になっています。有効にすると、次の要素が暗号化されます。

- 同期、バックアップ、およびスワップオペレーション中のディスク
- バックアップは Amazon S3 に保存されます。

データ階層化が有効なクラスター内の SSD (ソリッドステートドライブ) に保存されたデータは、常時暗号化されます。

ElastiCache では、保管時のデフォルト (サービス管理) の暗号化が用意されているだけでなく、お客様独自のカスタマー管理の対称 AWS KMS キーを [AWS Key Management Service \(KMS\)](#) で使用することもできます。キャッシュをバックアップするときに、暗号化オプションで、デフォルトの暗号化キーを使用するか、カスタマー管理のキーを使用するかを選択します。詳細については、「[保管時の暗号化を有効にする](#)」を参照してください。

Note

デフォルトの (サービス管理) 暗号化は、GovCloud (US) リージョンで使用できる唯一のオプションです。

Important

既存の独自設計型 Redis クラスターで保管時の暗号化を有効にするには、レプリケーショングループでバックアップと復元を実行した後で、既存のレプリケーショングループを削除する必要があります。

保管時の暗号化は、キャッシュに対してその作成時にのみ有効にできます。データの暗号化と復号を行うにはある程度の処理が必要であるため、保管時の暗号化を有効にすると、これらのオペレーショ

ンの実行中のパフォーマンスに影響を与える可能性があります。保管時の暗号化の使用時と未使用時でデータのベンチマークを取得して、ユースケースにおけるパフォーマンスの影響を判断する必要があります。

トピック

- [保管時の暗号化の条件](#)
- [AWS KMS のカスタマー管理のキーの使用](#)
- [保管時の暗号化を有効にする](#)
- [以下の資料も参照してください。](#)

保管時の暗号化の条件

ElastiCache の保管時の暗号化の実装を計画する際は、ElastiCache の保管時の暗号化の以下の制約事項に留意する必要があります。

- 保管時の暗号化は、Redis バージョン (EOL が予定されている 3.2.6、[「Redis バージョンのサポート終了スケジュール」](#)を参照)、4.0.10 またはそれ以降を実行しているレプリケーショングループでサポートされます。
- 保管時の暗号化は、Amazon VPC で実行されているレプリケーショングループでのみサポートされます。
- 保管時の暗号化は、以下のノードタイプを実行しているレプリケーショングループでのみサポートされます。
 - R6gd、R6g、R5、R4、R3
 - M6g、M5、M4、M3
 - T4g、T3、T2

詳細については、[サポートされているノードの種類](#)を参照してください。

- 保管時の暗号化は、パラメータ `AtRestEncryptionEnabled` を明示的に `true` に設定することで有効化されます。
- 保管時の暗号化は、レプリケーショングループの作成時にのみレプリケーショングループで有効にできます。レプリケーショングループを変更して保管時の暗号化のオンとオフを切り替えることはできません。既存のレプリケーショングループ上への保管時の暗号化の実装の詳細については、[「保管時の暗号化を有効にする」](#)を参照してください。
- クラスターが r6gd ファミリのノードタイプを使用している場合、保管時の暗号化が有効になっているかどうかにかかわらず、SSD に保存されているデータは暗号化されます。

- AWS GovCloud (us-gov-east-1 および us-gov-west-1) リージョンでは、保管時の暗号化にカスタマー管理のキーを使用することを選択できません。
- クラスターが r6gd ファミリーのノードタイプを使用している場合、SSD に保存されているデータは、選択したカスタマー管理の AWS KMS キー (または AWS GovCloud リージョンのサービスで管理された暗号化) で暗号化されます。

保管時の暗号化を実装することで、バックアップオペレーションおよびノード同期オペレーションの実行中にパフォーマンスが低下する場合があります。自身のデータで保管時の暗号化使用時のベンチマークを暗号化なしの場合と比較して、実装におけるパフォーマンスの影響を判断してください。

AWS KMS のカスタマー管理のキーの使用

ElastiCache は、保管時の暗号化用の対称カスタマー管理の AWS KMS キー (KMS キー) をサポートしています。カスタマー管理の KMS キーは、AWS アカウントで作成、所有、管理される暗号化キーです。詳細については、AWS Key Management Service デベロッパーガイドの「[AWS KMS キー](#)」を参照してください。キーは、ElastiCache で使用する前に AWS KMS で作成する必要があります。

AWS KMS ルートキーを作成する方法の詳細については、AWS Key Management Service デベロッパーガイドの「[キーの作成](#)」を参照してください。

ElastiCache を使用すると、AWS KMS と統合できます。詳細については、AWS Key Management Service デベロッパーガイドの「[付与の使用](#)」を参照してください。Amazon ElastiCache と AWS KMS の統合を有効にするために、お客様の作業は必要ありません。

kms:ViaService 条件キーは、AWS KMS キー (KMS キー) の使用を、指定の AWS サービスからのリクエストだけに制限します。ElastiCache とともに kms:ViaService を使用するには、elasticache.AWS_region.amazonaws.com と dax.AWS_region.amazonaws.com の条件キーの値に両方の ViaService 名を含めます。詳細については、「[kms:ViaService](#)」を参照してください。

[AWS CloudTrail](#) を使用して、Amazon ElastiCache によってお客様に代わって AWS Key Management Service に送信されるリクエストを追跡できます。カスタマー管理のキーに関連する AWS Key Management Service へのすべての API コールには、対応する CloudTrail ログがあります。[ListGrants](#) KMS API コールを行うことで、ElastiCache によって作成される許可を表示することもできます。

カスタマー管理のキーを使用してレプリケーショングループが暗号化されると、レプリケーショングループのすべてのバックアップは以下のように暗号化されます。

- 毎日の自動バックアップは、クラスターに関連付けられたカスタマー管理のキーを使用して暗号化されます。
- レプリケーショングループが削除されたときに作成される最終バックアップも、レプリケーショングループに関連付けられたカスタマー管理のキーを使用して暗号化されます。
- 手動で作成されたバックアップは、デフォルトで、レプリケーショングループに関連付けられた KMS キーを使用して暗号化されます。この動作は、別のカスタマー管理のキーを選択して上書きできます。
- バックアップをコピーするとき、デフォルトでは、ソースバックアップに関連付けられたカスタマー管理のキーが使用されます。この動作は、別のカスタマー管理のキーを選択して上書きできます。

Note

- 選択した Amazon S3 バケットにバックアップをエクスポートするとき、カスタマー管理のキーは使用できません。ただし、Amazon S3 にエクスポートされたすべてのバックアップは、[サーバー側の暗号化](#)を使用して暗号化されます。バックアップファイルを新しい S3 オブジェクトにコピーし、カスタマー管理の KMS キーを使用して暗号化するか、デフォルトの暗号化が設定された別の S3 バケットにコピーし、KMS キーを使用して暗号化するか、ファイル自体の暗号化オプションを変更できます。
- また、暗号化にカスタマー管理のキーを使用しないレプリケーショングループに手動で作成されたバックアップを、カスタマー管理のキーを使用して暗号化することもできます。このオプションでは、データが元のレプリケーショングループで暗号化されていなくても、Amazon S3 に保存されているバックアップファイルは KMS キーを使用して暗号化されます。

バックアップから復元するときは、新しいレプリケーショングループの作成時に使用できるものと同様の暗号化オプションから選択できます。

- キーを削除するか、キーを[無効化](#)して、キャッシュの暗号化に使用したキーの[許可を取り消す](#)と、キャッシュは回復不可能になります。つまり、ハードウェア障害の後に変更も回復もできなくなります。AWSルートキーは 7 日間以上の待機期間後にのみ KMS によって削除されます。キーが削除された後、別のカスタマー管理のキーを使用して、アーカイブ目的のバックアップを作成できます。

- 自動キー更新は AWS KMS ルートキーのプロパティを保持するため、お客様が ElastiCache データにアクセスできるかどうかには影響しません。暗号化された Amazon ElastiCache キャッシュは、手動キー更新をサポートしていないため、新しいルートキーの作成や古いキーへの参照の更新などを行うことはできません。詳細については、AWS Key Management Service デベロッパーガイドの「[AWS KMS キーのローテーション](#)」を参照してください
- KMS キーを使用して ElastiCache キャッシュを暗号化するには、キャッシュごとに 1 つの許可が必要です。この許可はキャッシュの有効期間を通じて使用されます。また、バックアップの作成中、バックアップごとに 1 つの許可が使用されます。この許可はバックアップの作成後に無効になります。
- AWS KMS の付与と制限の詳細については、AWS Key Management Service デベロッパーガイドの「[制限](#)」を参照してください。

保管時の暗号化を有効にする

すべてのサーバーレスキャッシュでは、保管時の暗号化が有効になっています。

独自設計型クラスターを作成する場合は、パラメータ `AtRestEncryptionEnabled` を `true` に設定することで保管時の暗号化を有効にできます。既存のレプリケーショングループ上で保管時の暗号化を有効にすることはできません。

保管時の暗号化は、ElastiCache キャッシュの作成時に有効化できます。これを行うには、AWS Management Console、AWS CLI、または ElastiCache API を使用します。

キャッシュを作成するときに、以下のオプションのいずれかを選択できます。

- デフォルト - このオプションでは、サービス管理の保存時の暗号化が使用されます。
- [カスタマー管理のキー] - このオプションでは、AWS KMS からのキー ID/ARN を保管時の暗号化に使用できます。

AWS KMS ルートキーを作成する方法については、AWS Key Management Service デベロッパーガイドの「[キーの作成](#)」を参照してください。

目次

- [AWS Management Console を使用して保管時の暗号化を有効にする](#)
- [AWS CLI を使用して保管時の暗号化を有効にする](#)

既存の独自設計型 Redis クラスター上で保管時の暗号化を有効にする

保管時の暗号化は、Redis レプリケーショングループの作成時にのみ有効化できます。保管時の暗号化を有効化したい既存レプリケーショングループがある場合は、次の操作を行います。

既存のレプリケーショングループ上で保管時の暗号化を有効にするには

1. 既存のレプリケーショングループの手動バックアップを作成します。詳細については、「[手動バックアップの取得](#)」を参照してください。
2. バックアップから復元して新しいレプリケーショングループを作成します。新しいレプリケーショングループで、保管時の暗号化を有効にします。詳細については、「[バックアップから新しいキャッシュへの復元](#)」を参照してください。
3. アプリケーションのエンドポイントを、新しいレプリケーショングループのエンドポイントに更新します。
4. 古いレプリケーショングループを削除します。詳細については、「[クラスターの削除](#)」または「[レプリケーショングループの削除](#)」を参照してください。

AWS Management Console を使用して保管時の暗号化を有効にする

サーバーレスキャッシュで保管時の暗号化を有効にする (コンソール)

すべてのサーバーレスキャッシュでは、保管時の暗号化が有効になっています。デフォルトでは、AWS 所有の KMS キーがデータを暗号化するために使用されます。独自の AWS KMS キーを選択するには、以下のように選択します。

- [デフォルト設定] セクションを展開します。
- [デフォルト設定] セクションで [デフォルト設定をカスタマイズ] を選択します。
- [セキュリティ] セクションで [セキュリティ設定をカスタマイズ] を選択します。
- [暗号化キー] 設定で [カスタマーマネージド CMK] を選択します。
- [AWS KMS キー] 設定でキーを選択します。

独自設計型クラスター上で保管時の暗号化を有効にする (コンソール)

独自のキャッシュを設計する場合、[簡易作成] 方式の [開発/テスト] 設定と [本番稼働用] 設定では、[デフォルト] キーを使用する保管時の暗号化が有効になっています。設定を自分で選択するときは、以下のように選択します。

- エンジンのバージョンとしてバージョン 3.2.6、4.0.10 またはそれ以降を選択します。

- [保管時の暗号化] オプションの [有効化] の横にあるチェックボックスをオンにします。
- [デフォルトキー] または [カスタマーマネージド CMK] を選択します。

詳しい手順については、以下を参照ください。

- [Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)
- [Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)

AWS CLI を使用して保管時の暗号化を有効にする

AWS CLI を使用して Redis クラスターを作成するときに保管時の暗号化を有効にするには、レプリケーショングループの作成時に `--at-rest-encryption-enabled` パラメーターを使用します。

Redis (クラスターモードが無効) クラスターで保管時の暗号化を有効にする (CLI)

次のオペレーションでは、3 つのノード (`--num-cache-clusters`)、1 つのプライマリ、2 つのリードレプリカを持つ Redis (クラスターモードが無効) レプリケーショングループ `my-classic-rg` を作成します。保管時の暗号化は、このレプリケーショングループに対して有効です (`--at-rest-encryption-enabled`)。

以下のパラメータとその値は、このレプリケーショングループで暗号化を有効にするために必要です。

主要パラメータ

- `--engine` — `redis` を指定する必要があります。
- `--engine-version` — 3.2.6 または 4.0.10 以降を指定する必要があります。
- `--at-rest-encryption-enabled` — 保管時の暗号化に必要です。

Example 1: レプリカがある Redis (クラスターモードが無効) クラスター

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id my-classic-rg \  
  --replication-group-description "3 node replication group" \  
  --cache-node-type cache.m4.large \  
  --engine redis \  
  --at-rest-encryption-enabled
```

```
--at-rest-encryption-enabled \  
--num-cache-clusters 3
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id my-classic-rg ^  
  --replication-group-description "3 node replication group" ^  
  --cache-node-type cache.m4.large ^  
  --engine redis ^  
  --at-rest-encryption-enabled ^  
  --num-cache-clusters 3 ^
```

詳細については、以下を参照してください。

- [Redis \(クラスターモードが無効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)
- [create-replication-group](#)

Redis (クラスターモードが有効) のクラスター上で保管時の暗号化を有効にする (CLI)

次のオペレーションでは、3 つのノードグループまたはシャード (--num-node-groups) を持つ Redis (クラスターモードが有効) レプリケーショングループ `my-clustered-rg` を作成します。各レプリケーショングループには、1 つのプライマリ、および 2 つのリードレプリカ (--replicas-per-node-group) があります。保管時の暗号化は、このレプリケーショングループに対して有効です (--at-rest-encryption-enabled)。

以下のパラメータとその値は、このレプリケーショングループで暗号化を有効にするために必要です。

主要パラメータ

- **--engine** — `redis` を指定する必要があります。
- **--engine-version** — 4.0.10 以降を指定する必要があります。
- **--at-rest-encryption-enabled** — 保管時の暗号化に必要です。
- **--cache-parameter-group** — `default-redis4.0.cluster.on`、またはこれをクラスターモードが有効なレプリケーショングループにするために、それから算出されたいずれかに指定する必要があります。

Example 2: Redis (クラスターモードが有効) クラスター

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id my-clustered-rg \  
  --replication-group-description "redis clustered cluster" \  
  --cache-node-type cache.m3.large \  
  --num-node-groups 3 \  
  --replicas-per-node-group 2 \  
  --engine redis \  
  --engine-version 6.2 \  
  --at-rest-encryption-enabled \  
  --cache-parameter-group default.redis6.x.cluster.on
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id my-clustered-rg ^  
  --replication-group-description "redis clustered cluster" ^  
  --cache-node-type cache.m3.large ^  
  --num-node-groups 3 ^  
  --replicas-per-node-group 2 ^  
  --engine redis ^  
  --engine-version 6.2 ^  
  --at-rest-encryption-enabled ^  
  --cache-parameter-group default.redis6.x.cluster.on
```

詳細については、以下を参照してください。

- [Redis \(クラスターモードが有効\) レプリケーショングループを最初から作成する \(AWS CLI\)](#)
- [create-replication-group](#)

以下の資料も参照してください。

- [Amazon VPC と ElastiCache のセキュリティ](#)
- [Amazon の Identity and Access Management ElastiCache](#)

認証と認可

ElastiCache では、IAM と Redis AUTH コマンドを使用したユーザーの認証と、ロールベースのアクセスコントロール (RBAC) を使用したユーザーの承認オペレーションをサポートします。

トピック

- [ロールベースのアクセスコントロール \(RBAC\)](#)
- [Redis AUTH コマンドによる認証](#)
- [ElastiCache Redis キャッシュでのアクセス制御の無効化](#)

ロールベースのアクセスコントロール (RBAC)

[Redis AUTH コマンドによる認証](#) で説明されているように、Redis 6.0 以降では、Redis AUTH コマンドを使用してユーザーを認証する代わりに、ロールベースのアクセス制御 (RBAC) と呼ばれる機能を使用できます。RBAC は、サーバーレスキャッシュへのアクセスを制御する唯一の手段でもあります。

Redis AUTH では、クライアントのトークンが認証されていれば、認証済みのすべてのクライアントにキャッシュへのフルアクセスが認められますが、RBAC はこれとは異なり、キャッシュへのアクセスをユーザーグループを通じて制御できます。ユーザーグループは、キャッシュへのアクセスを分類する手段として設計されています。

RBAC では、以下で説明されているように、アクセス文字列を使用してユーザーを作成し、ユーザーに特定のアクセス許可を割り当てます。特定のロール (管理者、人事) に該当するユーザーグループにユーザーを割り当てます。その後、それらのグループが 1 つ以上の ElastiCache for Redis キャッシュにデプロイされます。これにより、同じ Redis キャッシュを使用するクライアント間にセキュリティ境界を設け、クライアントが互いのデータにアクセスできないようにすることができます。

RBAC は、Redis 6 の [\[Redis ACL\]](#) の導入をサポートするように設計されています。ElastiCache for Redis キャッシュで RBAC を使用する場合は、制約がいくつかあります。

- アクセス文字列にパスワードを指定することはできません。パスワードは [CreateUser](#) または [ModifyUser](#) コールで設定します。
- ユーザー権限については、on および off をアクセス文字列の一部としてパスします。アクセス文字列にどちらも指定されていない場合、ユーザーには off が割り当てられ、キャッシュへのアクセス権はありません。

- 禁止されたコマンドや名前を変更したコマンドは使用できません。禁止または名前変更されたコマンドを指定すると、例外がスローされます。名前を変更したコマンドでアクセスコントロールリスト (ACL) を使用する場合は、コマンドの元の名前、つまり名前が変更される前のコマンドの名前を指定します。
- `reset` コマンドを、アクセス文字列の一部として使用することはできません。API パラメータを用いてパスワードを指定すると、ElastiCache for Redis がパスワードを管理します。したがって、`reset` を使用することはできません。それによりユーザーのすべてのパスワードが削除されるからです。
- Redis 6 は、[ACL リスト](#) コマンドを導入します。このコマンドは、ユーザーのリストと、各ユーザーに適用される ACL ルールを返します。ElastiCache for Redis は ACL LIST コマンドをサポートしますが、Redis のようにパスワードハッシュのサポートは含まれていません。ElastiCache for Redis を使用すると、[describe-users](#) オペレーションを使用して、アクセス文字列に含まれるルールなど、同様の情報を取得できます。ただし、[describe-users](#) は、ユーザーパスワードを取得しません。

ElastiCache for Redis でサポートされているその他の読み取り専用コマンドには、[ACL WHOAMI](#)、[ACL USERS](#)、[ACL CAT](#)などがあります。ElastiCache for Redis は、他の書き込みベースの ACL コマンドをサポートしていません。

- 以下の制約が適用されます。

[リソース]	最大許容数
ユーザーグループあたりのユーザー数	100
ユーザー数	1,000
ユーザーグループの数	100

ElastiCache for Redis での RBAC の使用については、以下で詳しく説明します。

トピック

- [アクセス文字列を使用したアクセス許可の指定](#)
- [ElastiCache for Redis のキャッシュへの RBAC の適用](#)
- [Redis AUTH から RBAC への移行](#)
- [RBAC から Redis AUTH への移行](#)

- [ユーザーのパスワードの自動ローテーション](#)
- [IAM を使用した認証](#)

アクセス文字列を使用したアクセス許可の指定

ElastiCache for Redis キャッシュへのアクセス許可を指定するには、アクセス文字列を作成し、AWS CLI または AWS Management Console を使用して、そのアクセス文字列をユーザーに割り当てます。

アクセス文字列は、ユーザーに適用されるスペース区切りルールの一覧として定義されます。それらは、ユーザーが実行できるコマンドと、ユーザーが操作できるキーを定義します。コマンドを実行するには、ユーザーは、実行されているコマンドと、そのコマンドによってアクセスされているすべてのキーにアクセスする必要があります。ルールは左から右に累積的に適用され、提供された文字列に冗長性がある場合は、提供された文字列の代わりに、より単純な文字列を使用できます。

ACL ルールの構文の詳細については、「[ACL](#)」を参照してください。

次の例では、アクセス文字列は、使用可能なすべてのキーおよびコマンドにアクセスできるアクティブなユーザーを表します。

```
on ~* +@all
```

アクセス文字列の構文は、次のように分類されます。

- `on` — ユーザーはアクティブなユーザーです。
- `~*` — アクセス権はすべての使用可能なキーに与えられます。
- `+@all` — アクセス権はすべての使用可能なコマンドに与えられます。

上記の設定は、最も制限が緩い設定です。これらの設定を変更して、セキュリティを強化できます。

次の例では、アクセス文字列は「`app::`」キースペースで始まるキーに対する読み取りアクセスに制限されたアクセス権を持つユーザーを表します。

```
on ~app::* -@all +@read
```

ユーザーがアクセス権を持つコマンドを一覧表示することで、これらのアクセス許可をさらに絞り込むことができます。

`+command1` — ユーザーのコマンドへのアクセスは `command1` に制限されます。

+@category — ユーザーのアクセスは、コマンドのカテゴリに制限されます。

アクセス文字列をユーザーに割り当てる方法については、「[コンソールと CLI を使用したユーザーおよびユーザーグループの作成](#)」を参照してください。

既存のワークロードを ElastiCache に移行する場合は、ACL LIST を呼び出すことでアクセス権を取得して、ユーザーおよびパスワードハッシュを除外できます。

Redis バージョン 6.2 以降では、以下のアクセス文字列構文もサポートされています。

- &* — アクセス権はすべての使用可能なチャンネルに与えられます。

Redis バージョン 7.0 以降では、以下のアクセス文字列構文もサポートされています。

- | — サブコマンドをブロックするために使用できます (例: 「-config|set」)。
- %R~<pattern> — 指定された読み取りキーパターンを追加します。これは通常のキーパターンと同様に動作しますが、指定されたパターンに一致するキーからの読み取り権限のみを許可します。詳細については、「[キーのアクセス許可](#)」を参照してください。
- %W~<pattern> — 指定された書き込みキーパターンを追加します。これは通常のキーパターンと同様に動作しますが、指定されたパターンに一致するキーに書き込む権限のみを許可します。詳細については、「[キーのアクセス許可](#)」を参照してください。
- %RW~<pattern> - ~<pattern> のエイリアス。
- (<rule list>) — ルールを照合する新しいセクターを作成します。セクターはユーザーアクセス許可の後に評価され、定義されている順序に従って評価されます。コマンドがユーザーアクセス許可または任意のセクターと一致する場合、そのコマンドは許可されます。詳細については、[ACL セクター](#)を参照してください。
- clearselectors — ユーザーにアタッチされているセクターをすべて削除します。

ElastiCache for Redis のキャッシュへの RBAC の適用

ElastiCache for Redis RBAC を使用するには、次のステップに従います。

1. 1 つ以上のユーザーを作成します。
2. ユーザーグループを作成し、ユーザーをグループに追加します。
3. 転送時の暗号化が有効なキャッシュにユーザーグループを割り当てます。

これらのステップは、以下に詳細が説明されます。

トピック

- [コンソールと CLI を使用したユーザーおよびユーザーグループの作成](#)
- [コンソールおよび CLI を使用したユーザーグループの管理](#)
- [サーバーレスキャッシュへのユーザーグループの割り当て](#)
- [レプリケーショングループへのユーザーグループの割り当て](#)

コンソールと CLI を使用したユーザーおよびユーザーグループの作成

RBAC ユーザーのユーザー情報は、ユーザー ID、ユーザー名、およびオプションのパスワードとアクセス文字列です。アクセス文字列は、キーとコマンドでのアクセス許可レベルを提供します。ユーザー ID はユーザーに対して一意であり、ユーザー名はエンジンに渡されるものです。

指定するユーザー許可が、ユーザーグループの意図した目的に合っていることを確認してください。たとえば、Administrators という名前のユーザーグループを作成した場合、そのグループに追加するすべてのユーザーは、キーおよびコマンドへのフルアクセスに設定されたアクセス文字列を持つ必要があります。e-commerce ユーザーグループ内のユーザーでは、アクセス文字列を読み取り専用アクセスに設定できます。

ElastiCache は、ユーザー ID とユーザー名 "default" を使用してデフォルトユーザーを自動的に設定し、それをすべてのユーザーグループに追加します。このユーザーを変更または削除することはできません。このユーザーは、以前の Redis バージョンのデフォルト動作との互換性を意図しており、すべてのコマンドを呼び出してすべてのキーにアクセスできるようにするアクセス文字列を持っています。

適切なアクセスコントロールをキャッシュに追加するには、このデフォルトユーザーを、有効になっていない、または強力なパスワードを使用する新しいユーザーに置き換えます。デフォルトユーザーを変更するには、ユーザー名が default に設定された新しいユーザーを作成します。その後、元のデフォルトユーザーと入れ替えることができます。

次の手順では、元の default ユーザーを、変更されたアクセス文字列を持つ別の default ユーザーと入れ替える方法を示します。

コンソールでデフォルトユーザーを変更するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで [ユーザーグループの管理] を選択します。

3. [ユーザーグループ ID] で、変更対象の ID を選択します。チェックボックスではなく、リンクを選択するようにしてください。
4. [Modify] (変更) を選択します。
5. [変更] ウィンドウで、[管理] を選択し、[ユーザー名] で、デフォルトユーザーとして使用したいユーザーを選択します。
6. [選択] を選択します。
7. [Modify] (変更) を選択します。これを行うと、元のデフォルトユーザーが確立していたキャッシュへの既存の接続はすべて終了します。

AWS CLI を使用してデフォルトユーザーを変更するには

1. 以下のコマンドを使用して、ユーザー名 default で新しいユーザーを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-user \  
  --user-id "new-default-user" \  
  --user-name "default" \  
  --engine "REDIS" \  
  --passwords "a-strong-password" \  
  --access-string "off +get ~keys*"
```

Windows の場合:

```
aws elasticache create-user ^  
  --user-id "new-default-user" ^  
  --user-name "default" ^  
  --engine "REDIS" ^  
  --passwords "a-strong-password" ^  
  --access-string "off +get ~keys*"
```

2. ユーザーグループを作成し、前に作成したユーザーを追加します。

Linux、macOS、Unix の場合:

```
aws elasticache create-user-group \  
  --user-group-id "new-group-2" \  
  --engine "REDIS" \  
  --user-ids "new-default-user"
```

Windows の場合:

```
aws elasticache create-user-group ^
  --user-group-id "new-group-2" ^
  --engine "REDIS" ^
  --user-ids "new-default-user"
```

3. 新しい default ユーザーを元の default ユーザーと入れ替えます。

Linux、macOS、Unix の場合:

```
aws elasticache modify-user-group \
  --user-group-id test-group \
  --user-ids-to-add "new-default-user" \
  --user-ids-to-remove "default"
```

Windows の場合:

```
aws elasticache modify-user-group ^
  --user-group-id test-group ^
  --user-ids-to-add "new-default-user" ^
  --user-ids-to-remove "default"
```

この変更オペレーションが呼び出されると、元のデフォルトユーザーが確立していたキャッシュへの既存の接続はすべて終了します。

ユーザーを作成するときは、最大 2 つのパスワードを設定できます。パスワードを変更しても、キャッシュへの既存の接続はすべて維持されます。

特に、ElastiCache for Redis で RBAC を使用する場合は、以下のユーザーパスワードの制約に注意してください。

- パスワードは、印刷可能な 16~128 文字にする必要があります。
- 次の英数字以外の文字は使用できません: , " / @。

コンソールおよび CLI を使用したユーザーの管理

コンソール上でユーザーを管理するには、次の手順に従います。

コンソールでユーザーを管理するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. Amazon ElastiCache ダッシュボードで、[ユーザー管理] を選択します。以下のオプションが利用できます。
 - [ユーザーを作成] — ユーザーの作成時に、ユーザー ID、ユーザー名、認証モード、およびアクセス文字列を入力します。アクセス文字列は、ユーザーが許可されたキーとコマンドのアクセス許可レベルを設定します。

ユーザーを作成するときは、最大 2 つのパスワードを設定できます。パスワードを変更しても、キャッシュへの既存の接続はすべて維持されます。

- [ユーザーを変更] — ユーザーの認証設定を更新したり、アクセス文字列を変更したりできます。
- [ユーザーを削除] — アカウントは、所属先のすべてのユーザーグループから削除されます。

AWS CLI でユーザーを管理するには、次の手順を使用します。

CLI を使用してユーザーを変更するには

- `modify-user` コマンドを使用して、ユーザーのパスワードまたはパスワードを更新したり、ユーザーのアクセス権を変更したりします。

ユーザーが変更されると、そのユーザーに関連付けられたユーザーグループが更新され、そのユーザーグループに関連付けられたキャッシュも更新されます。既存の接続はすべて維持されます。以下は例です。

Linux、macOS、Unix の場合:

```
aws elasticache modify-user \  
  --user-id user-id-1 \  
  --access-string "~objects:* ~items:* ~public:*" \  
  --no-password-required
```

Windows の場合:

```
aws elasticache modify-user ^  
  --user-id user-id-1 ^
```



```
--access-string "~objects:* ~items:* ~public:*" ^  
--no-password-required
```

Note

nopass オプションを使用することは推奨されません。その場合、ユーザーのアクセス許可を読み取り専用を設定して、限定されたキーのセットにアクセスすることをお勧めします。

CLI を使用してサービスロールを削除するには

- ユーザーを削除するには、`delete-user` コマンドを使用します。アカウントが削除され、そのアカウントが属するユーザーグループから削除されます。次に例を示します。

Linux、macOS、Unix の場合:

```
aws elasticache delete-user \  
--user-id user-id-2
```

Windows の場合:

```
aws elasticache delete-user ^  
--user-id user-id-2
```

ユーザーのリストを表示するには、[\[describe-users\]](#) オペレーションを呼び出します。

```
aws elasticache describe-users
```

コンソールおよび CLI を使用したユーザーグループの管理

次に示すように、ユーザーグループを作成して、1 つまたは複数のレプリケーショングループに対するユーザーのアクセスを分類および制御できます。


コンソールを使用してユーザーグループを管理するには、次の手順に従います。

コンソールを使用してユーザーグループを管理するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. Amazon ElastiCache ダッシュボードで、[ユーザーグループの管理] を選択します。

以下のオペレーションは、新しいユーザーグループを作成するために使用できます。

- [作成] – ユーザーグループの作成時に、ユーザーを追加し、ユーザーグループをキャッシュに割り当てます。例えば、キャッシュで管理者ロールを持つユーザーの Admin ユーザーグループを作成できます。


 Important

ユーザーグループを作成するときは、デフォルトユーザーを含める必要があります。

- [ユーザーの追加] – ユーザーをユーザーグループに追加します。
- [ユーザーの削除] – ユーザーグループからユーザーを削除します。ユーザーがユーザーグループから削除された場合、そのユーザーが確立しているキャッシュへの既存の接続はすべて終了します。
- [削除] – ユーザーグループを削除するには、これを使用します。グループに属するユーザーではなく、ユーザーグループ自体が削除されることに注意してください。

既存のユーザーグループでは、次のことを実行できます。

- [ユーザーの追加] – 既存のユーザーをユーザーグループに追加します。
- [ユーザーを削除する] – ユーザーグループから既存のユーザーを削除します。

 Note

ユーザーはユーザーグループから削除されますが、システムからは削除されません。

次の手順で、CLI を使用してユーザーグループを管理します。

CLI を使用して新しいユーザーグループを作成し、ユーザーを追加するには

- 次に示すように、`create-user-group` コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache create-user-group \  
  --user-group-id "new-group-1" \  
  --engine "REDIS" \  
  --user-ids user-id-1, user-id-2
```

Windows の場合:

```
aws elasticache create-user-group ^  
  --user-group-id "new-group-1" ^  
  --engine "REDIS" ^  
  --user-ids user-id-1, user-id-2
```

CLI を使用して新しいユーザーを追加するか、現在のメンバーを削除してユーザーグループを変更するには

- 次に示すように、`modify-user-group` コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-user-group --user-group-id new-group-1 \  
  --user-ids-to-add user-id-3 \  
  --user-ids-to-remove user-id-2
```

Windows の場合:

```
aws elasticache modify-user-group --user-group-id new-group-1 ^  
  --user-ids-to-add userid-3 ^  
  --user-ids-to-remove user-id-2
```

Note

ユーザーグループから削除されたユーザーに属する開いている接続はすべて、このコマンドによって終了します。

CLI を使用してターゲットグループを削除するには

- 次に示すように、`delete-user-group` コマンドを使用します。グループに属するユーザーではなく、ユーザーグループ自体が削除されます。

Linux、macOS、Unix の場合:

```
aws elasticache delete-user-group /  
  --user-group-id
```

Windows の場合:

```
aws elasticache delete-user-group ^  
  --user-group-id
```

ユーザーグループのリストを表示するには、[describe-user-groups](#) オペレーションを呼び出すことができます。

```
aws elasticache describe-user-groups \  
  --user-group-id test-group
```

サーバーレスキャッシュへのユーザーグループの割り当て

ユーザーグループを作成してユーザーを追加した後、RBAC の実装手順の締めくくりとして、ユーザーグループをサーバーレスキャッシュに割り当てます。

コンソールを使用したサーバーレスキャッシュへのユーザーグループの割り当て

AWS Management Console を使用してユーザーグループをサーバーレスキャッシュに追加するには、次の手順に従います。

- クラスタモードが無効の場合は、「[Redis \(クラスタモードが無効\) クラスタの作成 \(コンソール\)](#)」を参照してください。
- クラスタモードが有効の場合は、「[Redis \(クラスタモードが有効\) クラスタの作成 \(コンソール\)](#)」を参照してください。

AWS CLI を使用したサーバーレスキャッシュへのユーザーグループの割り当て

以下の AWS CLI オペレーションでは、`user-group-id` パラメータに値 *my-user-group-id* を指定して、サーバーレスキャッシュを作成します。サブネットグループ `sng-test` を、実存のサブネットグループに置き換えます。

主要パラメータ

- `--engine` — `redis` を指定する必要があります。
- `--user-group-id` — この値には、キャッシュへの特定のアクセス許可を割り当てられたユーザーで構成されるユーザーグループの ID を指定します。

Linux、macOS、Unix の場合:

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name "new-serverless-cache" \  
  --description "new-serverless-cache" \  
  --engine "redis" \  
  --user-group-id "new-group-1"
```

Windows の場合:

```
aws elasticache create-serverless-cache ^  
  --serverless-cache-name "new-serverless-cache" ^  
  --description "new-serverless-cache" ^  
  --engine "redis" ^  
  --user-group-id "new-group-1"
```

以下の AWS CLI オペレーションでは、`user-group-id` パラメータに値 *my-user-group-id* を指定して、サーバーレスキャッシュを変更します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-serverless-cache \  
  --serverless-cache-name serverless-cache-1 \  
  --user-group-id "new-group-2"
```

Windows の場合:

```
aws elasticache modify-serverless-cache ^
```

```
--serverless-cache-name serverless-cache-1 ^
--user-group-id "new-group-2"
```

キャッシュに加えられた変更は、非同期で更新されます。イベントを表示して、この進行状況をモニタリングできます。詳細については、「[ElastiCache イベントの表示](#)」を参照してください。

レプリケーショングループへのユーザーグループの割り当て

ユーザーグループを作成してユーザーを追加した後、RBAC を実装する最後の手順では、ユーザーグループをレプリケーショングループに割り当てます。

コンソールを使用したレプリケーショングループへのユーザーグループの割り当て

AWS Management Console を使用してユーザーグループをレプリケーションに追加するには、以下を行います。

- クラスターモードが無効の場合は、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」を参照してください。
- クラスターモードが有効の場合は、「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」を参照してください。

AWS CLI を使用したレプリケーショングループへのユーザーグループの割り当て

以下の AWS CLI オペレーションでは、転送時の暗号化 (TLS) が有効で、値 *my-user-group-id* を持つ `user-group-ids` パラメータを使用したレプリケーショングループを作成します。サブネットグループ `sng-test` を、実存のサブネットグループに置き換えます。

主要パラメータ

- `--engine` — `redis` を指定する必要があります。
- `--engine-version` - 6.0 以降を指定する必要があります。
- `--transit-encryption-enabled` — 認証およびユーザーグループの関連付けに必要です。
- `--user-group-ids` — この値には、キャッシュへの特定のアクセス許可を割り当てられたユーザーで構成されるユーザーグループの ID を指定します。
- `--cache-subnet-group` — ユーザーグループを関連付けるために必要です。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id "new-replication-group" \  
  --replication-group-description "new-replication-group" \  
  --engine "redis" \  
  --cache-node-type cache.m5.large \  
  --transit-encryption-enabled \  
  --user-group-ids "new-group-1" \  
  --cache-subnet-group "cache-subnet-group"
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id "new-replication-group" ^  
  --replication-group-description "new-replication-group" ^  
  --engine "redis" ^  
  --cache-node-type cache.m5.large ^  
  --transit-encryption-enabled ^  
  --user-group-ids "new-group-1" ^  
  --cache-subnet-group "cache-subnet-group"
```

以下の AWS CLI オペレーションでは、転送時の暗号化 (TLS) が有効で、値 *my-user-group-id* を持つ `user-group-ids` パラメータを使用したレプリケーショングループを変更します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id replication-group-1 \  
  --user-group-ids-to-remove "new-group-1" \  
  --user-group-ids-to-add "new-group-2"
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id replication-group-1 ^  
  --user-group-ids-to-remove "new-group-1" ^  
  --user-group-ids-to-add "new-group-2"
```

応答内の `PendingChanges` を書き留めます。キャッシュに加えられた変更は、非同期で更新されます。イベントを表示して、この進行状況をモニタリングできます。詳細については、「[ElastiCache イベントの表示](#)」を参照してください。

Redis AUTH から RBAC への移行

[Redis AUTH コマンドによる認証](#) で説明されているように Redis AUTH を使用していて、RBAC の使用に移行するには、次の手順を使用します。

次の手順を使用して、コンソールを使用して Redis AUTH から RBAC に移行します。

コンソールを使用して Redis AUTH から RBAC に移行するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 右上のリストから、変更対象のキャッシュが存在する AWS リージョンを選択します。
3. ナビゲーションペインで、変更対象のキャッシュで実行されているエンジンを選択します。

選択したエンジンのキャッシュのリストが表示されます。

4. キャッシュのリストで、変更対象のキャッシュの名前を選択します。
5. [アクション]、[変更] の順に選択します。
[変更] ウィンドウが表示されます。
6. [アクセスコントロール] で、[ユーザーグループのアクセスコントロールリスト] を選択します。
7. [ユーザーグループのアクセスコントロールリスト] で、ユーザーグループを選択します。
8. [変更をプレビュー] を選択し、次の画面で [変更] を選択します。

次の手順で、CLI を使用して Redis AUTH から RBAC に移行します。

CLI を使用して Redis AUTH から RBAC に移行するには

- 次に示すように、`modify-replication-group` コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group --replication-group-id test \  
--auth-token-update-strategy DELETE \  
--user-group-ids-to-add user-group-1
```

Windows の場合:


```
aws elasticache modify-replication-group --replication-group-id test ^
--auth-token-update-strategy DELETE ^
--user-group-ids-to-add user-group-1
```

RBAC から Redis AUTH への移行

RBAC を使用していて、Redis AUTH に移行する場合は、「[RBAC から Redis AUTH への移行](#)」を参照してください。

Note

ElastiCache キャッシュのアクセスコントロールを無効にする必要がある場合は、AWS CLI を使用して無効にする必要があります。詳細については、「[the section called “ElastiCache Redis キャッシュでのアクセス制御の無効化”](#)」を参照してください。

ユーザーのパスワードの自動ローテーション

AWS Secrets Manager を使用すると、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 コールで置き換えて、プログラムでシークレットを取得することができます。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、ElastiCache for Redis のパスワード (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 の詳細については、「[AWS Secrets Manager とは](#)」を参照してください。

ElastiCache がシークレットを使用する方法

Redis 6 では、Redis クラスターを保護するために ElastiCache for Redis で [ロールベースのアクセスコントロール \(RBAC\)](#) が導入されました。この機能により、実行できるコマンドとアクセスできるキーに関して特定の接続を制限できます。RBAC では、顧客がパスワードを使用してユーザーを作成するときに、パスワードの値はプレーンテキストで手動により入力する必要があり、オペレーターに表示されます。

シークレットマネージャーを使用すると、アプリケーションはパスワードを手動で入力してアプリケーションの設定に保存するのではなく、シークレットマネージャーからパスワードを取得します。これを行う方法については、「[ElastiCache ユーザーをシークレットに関連付ける方法](#)」を参照してください。

シークレットの使用にはコストが発生します。料金情報については、「[AWS Systems Manager の料金](#)」を参照してください。

ElastiCache ユーザーをシークレットに関連付ける方法

シークレットマネージャーは、関連するユーザーのリファレンスをシークレットの SecretString フィールドに保存します。ElastiCache 側からのシークレットへの参照はありません。

```
{
  "password": "strongpassword",
  "username": "user1",
  "user_arn": "arn:aws:elasticache:us-east-1:xxxxxxxxxxx918:user:user1" //this is the
  bond between the secret and the user
}
```

Lambda ローテーション関数

シークレットマネージャーの自動パスワードローテーションを有効にするには、[modify-user](#) API を操作してユーザーのパスワードを更新する Lambda 関数を作成します。

この仕組みについては、「[ローテーションの仕組み](#)」を参照してください。

Note

AWS はいくつかの AWS サービスにおいて、混乱した代理プログラムの問題を避けるために、「aws:SourceArn」と「aws:SourceAccount」のグローバルコンテキストキーを両方使用することを推奨しています。ただし、ローテーション関数のポリシーに aws:SourceArn の条件を含めると、その ARN で指定されたシークレットだけをローテーションさせるためにローテーション関数を使用することができます。コンテキストキーのみを含めることをお勧めします aws:SourceAccount 複数のシークレットに対して回転関数を使用できるようにする。

発生する可能性のある問題については、「[AWS シークレットマネージャーのローテーションのトラブルシューティング](#)」を参照してください。

ElastiCache ユーザーを作成してシークレットマネージャーに関連付ける方法

以下の手順は、ユーザーを作成してシークレットマネージャーに関連付ける方法を示しています。

1. 非アクティブユーザーの作成

Linux、macOS、Unix の場合:

```
aws elasticache create-user \  
  --user-id user1 \  
  --user-name user1 \  
  --engine "REDIS" \  
  --no-password \ // no authentication is required  
  --access-string "*off* +get ~keys*" // this disables the user
```

Windows の場合:

```
aws elasticache create-user ^  
  --user-id user1 ^  
  --user-name user1 ^  
  --engine "REDIS" ^  
  --no-password ^ // no authentication is required  
  --access-string "*off* +get ~keys*" // this disables the user
```

次のようなレスポンスが表示されます。

```
{  
  "UserId": "user1",  
  "UserName": "user1",  
  "Status": "active",  
  "Engine": "redis",  
  "AccessString": "off ~keys* -@all +get",  
  "UserGroupIds": [],  
  "Authentication": {  
    "Type": "no_password"  
  },  
  "ARN": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"  
}
```

2. シークレットを作成する

Linux、macOS、Unix の場合:

```
aws secretsmanager create-secret \  
--name production/ec/user1 \  
--secret-string \  
'{  
  "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",  
  "username": "user1"  
}'
```

Windows の場合:

```
aws secretsmanager create-secret ^  
--name production/ec/user1 ^  
--secret-string ^  
'{  
  "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",  
  "username": "user1"  
}'
```

次のようなレスポンスが表示されます。

```
{  
  "ARN": "arn:aws:secretsmanager:us-east-1:123456xxxx:secret:production/ec/user1-  
eaFois",  
  "Name": "production/ec/user1",  
  "VersionId": "aae5b963-1e6b-4250-91c6-ebd6c47d0d95"  
}
```

3. パスワードをローテーションするように Lambda 関数を設定する
 - a. AWS Management Console にサインインして <https://console.aws.amazon.com/lambda/> で Lambda コンソールを開きます。
 - b. ナビゲーションパネルで [Functions] (関数) を選択し、作成した関数を選択します。関数名の左側のチェックボックスではなく、関数名を選択します。
 - c. [設定] タブを選択します。
 - d. [General configuration] (一般設定) で、[Edit] (編集) を選択し、[Timeout] (タイムアウト) を 12 分以上に設定します。
 - e. [Save (保存)] を選択します。
 - f. [Environment variables] (環境変数) を選択し、以下を設定します。

- i. SECRETS_MANAGER_ENDPOINT – <https://secretsmanager.REGION.amazonaws.com>
 - ii. SECRET_ARN – ステップ 2 で作成したシークレットの Amazon リソースネーム (ARN)。
 - iii. USER_NAME – ElastiCache ユーザーのユーザー名、
 - iv. [Save (保存)] を選択します。
- g. [Permissions] (許可) を選択します。
 - h. [Execution role] (実行ロール) で、IAM コンソールに表示する Lambda 関数ロールの名前を選択します。
 - i. Lambda 関数でユーザーを変更してパスワードを設定するには、次のアクセス許可が必要です。

ElastiCache

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:DescribeUsers",
        "elasticache:ModifyUser"
      ],
      "Resource": "arn:aws:elasticache:us-east-1:xxxxxxxxxxx918:user:user1"
    }
  ]
}
```

Secrets Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:PutSecretValue",
        "secretsmanager:UpdateSecretVersionStage"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "arn:aws:secretsmanager:us-
east-1:xxxxxxxxxxxx:secret:XXXX"
  },
  {
    "Effect": "Allow",
    "Action": "secretsmanager:GetRandomPassword",
    "Resource": "*"
  }
]
```

4. シークレットマネージャーのシークレットローテーションを設定する

- a. AWS Management Console の使用方法については、「[コンソールを使用して AWS シークレットマネージャーのシークレットの自動ローテーションを設定する](#)」を参照してください

ローテーションのスケジュール設定の詳細については、「[シークレットマネージャーのローテーションでのスケジュール式](#)」を参照してください。

- b. AWS CLI の使用方法については、「[AWS Command Line Interface を使用して AWS Secrets Manager の自動ローテーションを設定する](#)」を参照してください

IAM を使用した認証

トピック

- [概要](#)
- [制限事項](#)
- [設定](#)
- [接続中](#)

概要

キャッシュが Redis バージョン 7 以上を使用するように設定されている場合、IAM 認証では、AWS IAM アイデンティティを使用して ElastiCache for Redis への接続を認証できます。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。また、IAM 認証を使用すると、個々の ElastiCache キャッシュと ElastiCache ユーザーごとにきめ細かいアクセス制御を設定し、最小特権の権限の原則に従うことができます。ElastiCache for Redis の IAM 認証は、有効期間の長い ElastiCache ユーザーパスワードの代わりに、有効期間が短い IAM 認証トークンを

Redis AUTH または HELLO コマンドで提供することにより機能します。IAM 認証トークンの詳細については、AWS 全般リファレンスガイドの「[Signature Version 4 signing process](#)」(署名バージョン 4 の署名プロセス)と、以下のコード例を参照してください。

IAM アイデンティティとそれに関連するポリシーを使用して、Redis アクセスをさらに制限できます。また、フェデレーテッド ID プロバイダーのユーザーに Redis キャッシュへのアクセス権を直接付与することもできます。

ElastiCache for Redis でAWS IAM を使用するには、まず認証モードを IAM に設定して ElastiCache ユーザーを作成する必要があります。その後で IAM アイデンティティを作成または再利用できます。IAM アイデンティティには、ElastiCache キャッシュと ElastiCache ユーザーに `elasticache:Connect` アクションを許可するための関連ポリシーが必要です。設定が完了すると、IAM ユーザーまたはロールの AWS 認証情報を使用して IAM 認証トークンを作成できます。最後に、Redis キャッシュに接続するときに、有効期間が短い IAM 認証トークンを Redis クライアントのパスワードとして指定する必要があります。認証情報プロバイダーをサポートしている Redis クライアントは、新しい接続ごとに一時的な認証情報を自動的に生成できます。ElastiCache for Redis は、IAM が有効な ElastiCache ユーザーの接続リクエストに対して IAM 認証を実行し、その接続リクエストを IAM で検証します。

制限事項

IAM 認証を使用する場合、以下の制限が適用されます。

- ElastiCache for Redis バージョン 7.0 以上を使用している場合、IAM 認証が利用できます。
- IAM が有効な ElastiCache ユーザーの場合、ユーザー名とユーザー ID のプロパティは同じである必要があります。
- IAM 認証トークンは 15 分間有効です。長時間接続する場合は、認証情報プロバイダーインターフェイスをサポートする Redis クライアントを使用することをお勧めします。
- ElastiCache for Redis への IAM 認証された接続は、12 時間後に自動的に切断されます。新しい IAM 認証トークンを使用して AUTH または HELLO コマンドを送信することで、接続を 12 時間延長できます。
- IAM 認証は MULTI EXEC コマンドではサポートされていません。
- 現在、IAM 認証は以下のグローバル条件コンテキストキーをサポートしています。
 - サーバーレスキャッシュで IAM 認証を使用する場合、`aws:VpcSourceIp`、`aws:SourceVpc`、`aws:SourceVpce`、`aws:CurrentTime`、`aws:EpochT%s` (関連するサーバーレスキャッシュとユーザーから) がサポートされます。

- レプリケーショングループで IAM 認証を使用する場合、aws:SourceIp および aws:ResourceTag/%s (関連するレプリケーショングループとユーザーから) がサポートされます。

グローバル条件コンテキストキーの詳細については、「IAM ユーザーガイド」の「[AWS グローバル条件コンテキストキー](#)」を参照してください。

設定

IAM 認証をセットアップするには:

1. キャッシュを作成します。

```
aws elasticache create-serverless-cache \  
  --serverless-cache-name cache-01 \  
  --description "ElastiCache IAM auth application" \  
  --engine redis
```

2. アカウントが新しいロールを引き継ぐことを許可するロール用の IAM 信頼ポリシードキュメントを以下に示すように作成します。ポリシーを trust-policy.json というファイルに保存します。

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },  
    "Action": "sts:AssumeRole"  
  }  
}
```

3. 以下に示すように、IAM ポリシードキュメントを作成します。ポリシーを policy.json というファイルに保存します。

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



```
    "Resource" : [  
      "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",  
      "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"  
    ]  
  }  
]  
}
```

4. IAM ロールを作成します。

```
aws iam create-role \  
--role-name "elasticache-iam-auth-app" \  
--assume-role-policy-document file://trust-policy.json
```

5. IAM ポリシーを作成します。

```
aws iam create-policy \  
--policy-name "elasticache-allow-all" \  
--policy-document file://policy.json
```

6. IAM ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \  
--role-name "elasticache-iam-auth-app" \  
--policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

7. IAM を有効にしている新しいユーザーを作成します。

```
aws elasticache create-user \  
--user-name iam-user-01 \  
--user-id iam-user-01 \  
--authentication-mode Type=iam \  
--engine redis \  
--access-string "on ~* +@all"
```

8. ユーザーグループを作成し、ユーザーをアタッチします。

```
aws elasticache create-user-group \  
--user-group-id iam-user-group-01 \  
--engine redis \  
--user-ids default iam-user-01  
  
aws elasticache modify-serverless-cache \  

```

```
--serverless-cache-name cache-01 \  
--user-group-id iam-user-group-01
```

接続中

トークンをパスワードとして接続

最初に、[AWS SigV4 の署名済みリクエスト](#)を使用して、有効期間が短い IAM 認証トークンを生成する必要があります。その後、以下の例に示すように、Redis キャッシュに接続するときに IAM 認証トークンをパスワードとして指定します。

```
String userId = "insert user id";  
String cacheName = "insert cache name";  
boolean isServerless = true;  
String region = "insert region";  
  
// Create a default AWS Credentials provider.  
// This will look for AWS credentials defined in environment variables or system  
// properties.  
AWSCredentialsProvider awsCredentialsProvider = new  
    DefaultAWSCredentialsProviderChain();  
  
// Create an IAM authentication token request and signed it using the AWS credentials.  
// The pre-signed request URL is used as an IAM authentication token for ElastiCache  
// Redis.  
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,  
    region, isServerless);  
String iamAuthToken =  
    iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());  
  
// Construct Redis URL with IAM Auth credentials provider  
RedisURI redisURI = RedisURI.builder()  
    .withHost(host)  
    .withPort(port)  
    .withSsl(ssl)  
    .withAuthentication(userId, iamAuthToken)  
    .build();  
  
// Create a new Lettuce Redis client  
RedisClient client = RedisClient.create(redisURI);  
client.connect();
```

以下は IAMAuthTokenRequest の定義です。

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
    private static final String PARAM_USER = "User";
    private static final String PARAM_RESOURCE_TYPE = "ResourceType";
    private static final String RESOURCE_TYPE_SERVERLESS_CACHE = "ServerlessCache";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "elasticache";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userId;
    private final String cacheName;
    private final String region;
    private final boolean isServerless;

    public IAMAuthTokenRequest(String userId, String cacheName, String region, boolean
isServerless) {
        this.userId = userId;
        this.cacheName = cacheName;
        this.region = region;
        this.isServerless = isServerless;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userId));
        if (isServerless) {
```

```
        request.addParameters(PARAM_RESOURCE_TYPE,
Collections.singletonList(RESOURCE_TYPE_SERVERLESS_CACHE));
    }
    return request;
}

private URI getRequestUri() {
    return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, cacheName));
}

private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
    AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(SERVICE_NAME);

    DateTime dateTime = DateTime.now();
    dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

    signer.presignRequest(request, credentials, dateTime.toDate());
}

private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
    return in.entrySet().stream()
        .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
        .collect(Collectors.toList());
}
}
```

認証情報プロバイダーに接続

以下のコードは、IAM 認証情報プロバイダーを使用して ElastiCache for Redis で認証する方法を示しています。

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
properties.
AWSCredentialsProvider awsCredentialsProvider = new
    DefaultAWSCredentialsProviderChain();
```

```
// Create an IAM authentication token request. Once this request is signed it can be
used as an
// IAM authentication token for ElastiCache Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
    region, isServerless);

// Create a Redis credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
    RedisIAMAuthCredentialsProvider(
        userId, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce Redis client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

以下は、IAMAuthTokenRequest を認証情報プロバイダーにラップして、必要に応じて一時的な認証情報を自動生成する Lettuce Redis クライアントの例です。

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userId;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userId,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userId;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
    }
}
```

```
        this.iamAuthTokenSupplier =  
        Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,  
        TimeUnit.SECONDS);  
    }  
  
    @Override  
    public Mono<RedisCredentials> resolveCredentials() {  
        return Mono.just(RedisCredentials.just(userId, iamAuthTokenSupplier.get()));  
    }  
  
    private String getIamAuthToken() {  
        return  
        iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());  
    }  
}
```

Redis AUTH コマンドによる認証

Note

Redis AUTHは に置き換えられました[the section called “ロールベースのアクセスコントロール \(RBAC\)”](#)。すべてのサーバーレスキャッシュでは、認証時に RBAC を使用する必要があります。

Redis 認証トークンまたはパスワードにより、Redis はクライアントにコマンドの実行を許可する前にパスワードを要求できるため、データセキュリティが向上します。Redis AUTHは独自設計型クラスターでのみ使用できます。

トピック

- [ElastiCache for Redis の AUTH の概要](#)
- [ElastiCache for Redis クラスターへの認証の適用](#)
- [既存の ElastiCache for Redis クラスターの AUTH トークンの変更](#)
- [RBAC から Redis AUTH への移行](#)

ElastiCache for Redis の AUTH の概要

ElastiCache for Redis クラスターAUTHで Redis を使用する場合、いくつかの改良点があります。

特に、for Redis で AUTH を使用する場合は、次の AUTH トークンまたはパスワード ElastiCache の制約に注意してください。

- トークン、またはパスワードは、印刷可能な 16 ~ 128 文字である必要があります。
- 英数字以外の文字は、(!、&、#、\$、^、<、>、-) に制限されています。
- AUTH は、Redis クラスターで転送中の暗号化 ElastiCache が有効になっている場合にのみ有効にできます。

強力なトークンを設定するには、以下の要件を満たすなど、厳格なパスワードポリシーに従うことをお勧めします。

- トークンまたはパスワードには、次の文字タイプのうち少なくとも 3 つを含める必要があります。
 - 英大文字
 - 英小文字
 - 数字
 - アルファベット以外の文字 (!、&、#、\$、^、<、>、-)
- トークンまたはパスワードには、ディクショナリワードまたはわずかに変更されたディクショナリワードを含めることはできません。
- トークンまたはパスワードは、最近使用したトークンと同じ、または類似するものであってはなりません。

ElastiCache for Redis クラスターへの認証の適用

トークン (パスワード) で保護された Redis サーバーでユーザーにトークンの入力を要求できます。そのためには、レプリケーショングループまたはクラスターを作成するときに、正しいトークンを指定したパラメータ `--auth-token` (API: `AuthToken`) を含めます。また、レプリケーショングループまたはクラスターに対する後続のすべてのコマンドにもそのパラメータを含めます。

次の AWS CLI オペレーションでは、転送時の暗号化 (TLS) が有効で、AUTH トークンを持つレプリケーショングループを作成します *This-is-a-sample-token*。サブネットグループ `sng-test` を、実存のサブネットグループに置き換えます。

キーのパラメータ

- `--engine` — `redis` を指定する必要があります。

- **--engine-version** — 3.2.6 または 4.0.10 以降を指定する必要があります。
- **--transit-encryption-enabled** — 認証と HIPAA 適格性に必要です。
- **--auth-token** — HIPAA 適格性に必要です。この値は、このトークンで保護された Redis サーバーの正しいトークンであることが必要です。
- **--cache-subnet-group** — HIPAA 適格性に必要です。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id authtestgroup \  
  --replication-group-description authtest \  
  --engine redis \  
  --cache-node-type cache.m4.large \  
  --num-node-groups 1 \  
  --replicas-per-node-group 2 \  
  --transit-encryption-enabled \  
  --auth-token This-is-a-sample-token \  
  --cache-subnet-group sng-test
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id authtestgroup ^  
  --replication-group-description authtest ^  
  --engine redis ^  
  --cache-node-type cache.m4.large ^  
  --num-node-groups 1 ^  
  --replicas-per-node-group 2 ^  
  --transit-encryption-enabled ^  
  --auth-token This-is-a-sample-token ^  
  --cache-subnet-group sng-test
```

既存の ElastiCache for Redis クラスターの AUTH トークンの変更

認証の更新を容易にするために、ElastiCache for Redis クラスターで使用される AUTH トークンを変更できます。エンジンバージョンが 5.0.6 以降で、ElastiCache for Redis で転送中の暗号化が有効になっている場合は、この変更を行うことができます。

認証トークンの変更は、ROTATE と SET の 2 つの戦略をサポートしています。ROTATE 戦略は、以前のトークンを保持しながら、サーバーに AUTH トークンを追加します。SET 戦略は、単一の

AUTH トークンのみをサポートするようにサーバーを更新します。変更をすぐに適用するには、これらの変更の呼び出しで `--apply-immediately` パラメータを指定します。

AUTH トークンの更新

Redis サーバーを新しい AUTH トークン で更新するには、`--auth-token` パラメータを新しい AUTH トークンとして `ModifyReplicationGroup` API を呼び出し、値を `ROTATE` `--auth-token-update-strategy` として呼び出します。ROTATE の変更が完了すると、クラスターは `auth-token` パラメータで指定されたトークンに加えて、以前の AUTH トークンをサポートします。AUTH トークンのローテーション前にレプリケーショングループに AUTH トークンが設定されていない場合、クラスターは認証なしの接続をサポートするだけでなく、`--auth-token` パラメータで指定された AUTH トークンもサポートします。更新戦略 SET [AUTH トークンの設定](#) を使用して AUTH トークンを必須に更新するには、「」を参照してください。

Note

AUTH トークンを前もって設定しない場合、変更が完了すると、クラスターは `auth-token` パラメータで指定されたもの以外の AUTH トークンをサポートしません。

この変更が既に 2 つの AUTH トークンをサポートしているサーバーで実行された場合、このオペレーション中に最も古い AUTH トークンも削除されます。これにより、サーバーは一度に最大 2 つの最新の AUTH トークンをサポートできます。

この時点で、最新の AUTH トークンを使用するようにクライアントを更新することで続行できます。クライアントが更新された後、AUTH トークンの更新に関する SET 戦略 (次のセクションで説明) を採用して、新しいトークンのみの使用を開始できます。

次の AWS CLI オペレーションでは、AUTH トークン をローテーションするようにレプリケーショングループを変更します ***This-is-the-rotated-token***。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
--replication-group-id authtestgroup \  
--auth-token This-is-the-rotated-token \  
--auth-token-update-strategy ROTATE \  
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-rotated-token ^
--auth-token-update-strategy ROTATE ^
--apply-immediately
```

AUTH トークンの設定

1 つの必須 AUTH トークンをサポートするように Redis サーバーを更新するには、最後の AUTH トークンと同じ値を持つ `--auth-token` パラメータと値を持つ `--auth-token-update-strategy` パラメータを使用して `ModifyReplicationGroup` API オペレーションを呼び出します。SET 戦略は、以前に ROTATE 戦略を使用した から 2 つの AUTH トークンまたは 1 つのオプションの AUTH トークンを持つクラスターでのみ使用できます。変更が完了すると、Redis サーバーは `auth-token` パラメータで指定された AUTH トークンのみをサポートします。

次の AWS CLI オペレーションでは、AUTH トークンを に設定するようにレプリケーショングループを変更します *This-is-the-set-token*。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
--replication-group-id authtestgroup \  
--auth-token This-is-the-set-token \  
--auth-token-update-strategy SET \  
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
--replication-group-id authtestgroup ^  
--auth-token This-is-the-set-token ^  
--auth-token-update-strategy SET ^  
--apply-immediately
```

既存の ElastiCache for Redis クラスターでの認証の有効化

既存の Redis サーバーで認証を有効にするには、`ModifyReplicationGroup` API オペレーションを呼び出します。 `--auth-token` パラメータで新しいトークンを指定し、 `--auth-token-update-strategy` パラメータで値 ROTATE を指定して、 `ModifyReplicationGroup` を呼び出します。

ROTATE の変更が完了すると、クラスターは認証なしの接続をサポートするだけでなく、`--auth-token` パラメータで指定された AUTH トークンもサポートします。すべてのクライアントアプリケーションが更新され、AUTH トークンで Redis に認証されたら、SET 戦略を使用して AUTH トークンを必要に応じてマークします。認証の有効化は、転送中の暗号化 (TLS) が有効な Redis サーバーでのみサポートされます。

RBAC から Redis AUTH への移行

で説明されているように Redis ロールベースのアクセスコントロール (RBAC) を使用してユーザーを認証していて [ロールベースのアクセスコントロール \(RBAC\)](#)、Redis AUTH に移行する場合は、次の手順を使用します。コンソールまたは CLI を使用して移行できます。

コンソールを使用して RBAC から Redis AUTH に移行するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/elasticache/> で ElastiCache コンソールを開きます。
2. 右上隅のリストから、変更するクラスターがある AWS リージョンを選択します。
3. ナビゲーションペインで、変更するクラスターで実行されているエンジンを選択します。

選択したエンジンのクラスターが一覧表示されます。

4. クラスターのリストで、変更するクラスターの名前を選択します。
5. [アクション]、[変更] の順に選択します。

[変更] ウィンドウが表示されます。

6. [アクセスコントロール] で、[Redis AUTH デフォルトユーザーアクセス] を選択します。
7. [Redis 認証トークン] で、新しいトークンを設定します。
8. [変更をプレビュー] を選択し、次の画面で [変更] を選択します。

を使用して RBAC から Redis AUTH に移行するには AWS CLI

次のいずれかのコマンドを使用して、Redis レプリケーショングループの新しいオプション AUTH トークンを設定します。オプションの Auth トークンは、SET 次のステップの更新戦略を使用して、Auth トークンが必須としてマークされるまで、レプリケーショングループへの認証されていないアクセスを許可することに注意してください。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \
```

```
--replication-group-id test \  
--remove-user-groups \  
--auth-token This-is-a-sample-token \  
--auth-token-update-strategy ROTATE \  
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
--replication-group-id test ^  
--remove-user-groups ^  
--auth-token This-is-a-sample-token ^  
--auth-token-update-strategy ROTATE ^  
--apply-immediately
```

上記のコマンドを実行した後、新しく設定されたオプションの AUTH トークンを使用して、Redis アプリケーションを更新して ElastiCache レプリケーショングループに認証できます。認証トークンのローテーションを完了するには、次のコマンド SET で更新戦略を使用します。これにより、必要に応じてオプションの AUTH トークンにマークされます。認証トークンの更新が完了すると、レプリケーショングループのステータスは と表示ACTIVEされ、このレプリケーショングループへのすべての Redis 接続には認証が必要です。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
--replication-group-id test \  
--auth-token This-is-a-sample-token \  
--auth-token-update-strategy SET \  
--apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
--replication-group-id test ^  
--remove-user-groups ^  
--auth-token This-is-a-sample-token ^  
--auth-token-update-strategy SET ^  
--apply-immediately
```

詳細については、「[Redis AUTH コマンドによる認証](#)」を参照してください。

Note

ElastiCache クラスターのアクセスコントロールを無効にする必要がある場合は、「」を参照してください [the section called “ElastiCache Redis キャッシュでのアクセス制御の無効化”](#)。

ElastiCache Redis キャッシュでのアクセス制御の無効化

以下の手順に従って、Redis TLS 対応キャッシュのアクセス制御を無効にします。Redis キャッシュには、Redis AUTH のデフォルトユーザーアクセスまたはユーザーグループアクセス制御リスト (RBAC) の 2 種類の構成のいずれかがあります。キャッシュが AUTH 構成で作成された場合は、ユーザーグループを削除してキャッシュを無効にする前に、RBAC 構成に変更する必要があります。キャッシュが RBAC 構成で作成されている場合は、そのまま無効にできます。

RBAC で構成された Redis サーバーレスキャッシュを無効にするには

1. ユーザーグループを削除してアクセスコントロールを無効にします。

```
aws elasticache modify-serverless-cache --serverless-cache-name <serverless-cache>
--remove-user-group
```

2. (オプション) サーバーレスキャッシュにユーザーグループが関連付けられていないことを確認します。

```
aws elasticache describe-serverless-caches --serverless-cache-name <serverless-cache>
{
  "...
  "UserGroupId": ""
  "...
}
```

AUTH トークンで構成された Redis キャッシュを無効にするには

1. AUTH トークンを RBAC に変更し、追加するユーザーグループを指定します。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-id-value> --auth-token-update-strategy DELETE --user-group-ids-to-add <user-group-value>
```

2. AUTH トークンが無効になり、ユーザーグループが追加されたことを確認します。

```
aws elasticache describe-replication-groups --replication-group-id <replication-group-id-value>
{
  "...",
  "AuthTokenEnabled": false,
  "UserGroupIds": [
    "<user-group-value>"
  ]
  "...",
}
```

3. ユーザーグループを削除してアクセスコントロールを無効にします。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-value> --user-group-ids-to-remove <user-group-value>
{
  "...",
  "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
  }
  "...",
}
```

4. (オプション) クラスターにユーザーグループが関連付けられていないことを確認します。AuthTokenEnabled フィールドも false と表示されるはずです。

```
aws elasticache describe-replication-groups --replication-group-id <replication-group-value>
"AuthTokenEnabled": false
```

RBAC で構成された Redis クラスターを無効にするには

1. ユーザーグループを削除してアクセスコントロールを無効にします。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-value> --user-group-ids-to-remove <user-group-value>
{
  "...
  "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
  }
  "...
}
```

2. (オプション) クラスターにユーザーグループが関連付けられていないことを確認します。AuthTokenEnabled フィールドも false と表示されるはずですが。

```
aws elasticache describe-replication-groups --replication-group-id <replication-group-value>
"AuthTokenEnabled": false
```

インターネットトラフィックのプライバシー

Amazon ElastiCache では、以下の方法によりキャッシュデータを不正なアクセスからセキュリティで保護します。

- [Amazon VPC と ElastiCache のセキュリティ](#) では、インストールに必要なセキュリティグループのタイプを説明します。
- [Amazon の Identity and Access Management ElastiCache](#) は、ユーザー、グループ、グループ、ロールの付与と制限のためのものです。

Amazon VPC と ElastiCache のセキュリティ

データのセキュリティは重要であるため、ElastiCache には、データへのアクセス権限を持つユーザーを制御するための手段が用意されています。データへのアクセスを制御する方法は、Amazon

Virtual Private Cloud (Amazon VPC) または Amazon EC2-Classic でクラスターを起動したかどうかによって決まります。

⚠ Important

ElastiCache クラスターの起動のために Amazon EC2-Classic の使用を非推奨にしました。現行世代のすべてのノードは Amazon Virtual Private Cloud のみで起動されます。

Amazon Virtual Private Cloud (Amazon VPC) サービスは、従来のデータセンターに非常によく似た仮想ネットワークを定義します。お客様が Amazon VPC を設定すると、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能になります。仮想ネットワークにキャッシュクラスターを追加でき、Amazon VPC のセキュリティグループを使用して、キャッシュクラスターへのアクセスを制御できます。

このセクションでは、Amazon VPC 内で手動で ElastiCache クラスターを設定する方法を説明します。この情報は、ElastiCache と Amazon VPC との連携について理解を深めたいユーザーを対象としています。

トピック

- [ElastiCache と Amazon VPC について理解する](#)
- [Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン](#)
- [Virtual Private Cloud \(VPC\) の作成](#)
- [Amazon VPC で実行されるキャッシュへの接続](#)

ElastiCache と Amazon VPC について理解する

ElastiCache は Amazon Virtual Private Cloud (Amazon VPC) と完全に統合されています。ElastiCache ユーザーにとって、これは次のことを意味します。

- AWS アカウントが EC2-VPC プラットフォームのみをサポートしている場合、ElastiCache は常に Amazon VPC でクラスターを起動します。
- AWS を初めて利用する場合、クラスターは Amazon VPC にデプロイされます。デフォルト VPC が自動的に作成されます。
- デフォルト VPC をお持ちのお客様が、クラスター起動時にサブネットを指定しなかった場合は、そのクラスターはお客様のデフォルト Amazon VPC で起動されます。

詳細については、「[サポートされているプラットフォームとデフォルト VPC があるかどうかを確認する](#)」を参照してください。

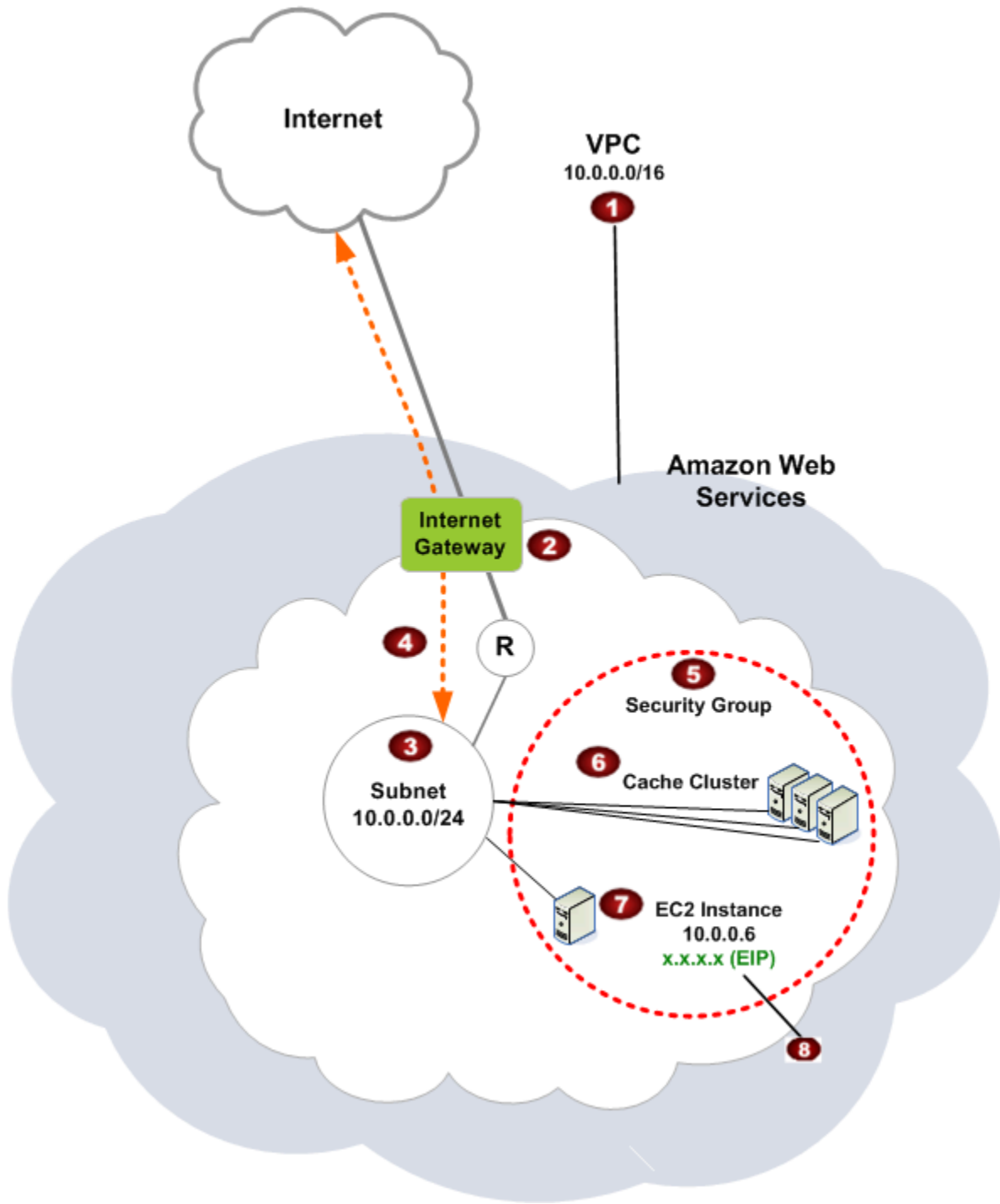
Amazon Virtual Private Cloud を使用することによって、従来のデータセンターに非常によく似た仮想ネットワークを AWS クラウド内に作成できます。お客様の Amazon VPC はお客様が設定できます。たとえば、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能です。

ElastiCache の基本機能は仮想プライベートクラウドの場合と同じです。ElastiCache は、クラスターが Amazon VPC の内部と外部のどちらにデプロイされるかに関係なく、ソフトウェアのアップグレード、パッチ適用、障害検出、および復旧を管理します。

Amazon VPC の外部にデプロイされた ElastiCache キャッシュノードには、エンドポイント/DNS 名の解決先となる IP アドレスが割り当てられます。これは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスからの接続を提供します。Amazon VPC プライベートサブネットで ElastiCache クラスターを起動した場合、すべてのキャッシュノードにはそのサブネット内のプライベート IP アドレスが割り当てられます。

Amazon VPC での ElastiCache の概要

次の図と表では、Amazon VPC 環境と、Amazon VPC で起動される ElastiCache クラスターと Amazon EC2 インスタンスについて説明します。

**1**

Amazon VPC は、独自の IP アドレスのブロックが割り当てられた AWS クラウドの独立した部分です。

2

インターネットゲートウェイは Amazon VPC を直接インターネットに接続し、他の AWS リソースへのアクセスを提供します。それには、Amazon VPC の外部で実行されている Amazon Simple Storage Service (Amazon S3) などのリソースが含まれます。

3 Amazon VPC サブネットは、セキュリティおよび運用上のニーズに合わせて AWS リソースを分離できる Amazon VPC の IP アドレス範囲のセグメントです。

4 Amazon VPC のルーティングテーブルは、サブネットとインターネットとの間でネットワークトラフィックを送信します。Amazon VPC には暗黙のルーターがあり、この図では円と R で表されています。

5 Amazon VPC セキュリティグループは、ElastiCache クラスターと Amazon EC2 インスタンスのインバウンドとアウトバウンドのトラフィックを制御します。

6 サブネットで ElastiCache クラスターを起動できます。キャッシュノードは、サブネットのアドレス範囲のプライベート IP アドレスを持ちます。

7 サブネットで Amazon EC2 インスタンスを起動することもできます。各 Amazon EC2 インスタンスはサブネットのアドレス範囲内のプライベート IP アドレスを持ちます。Amazon EC2 インスタンスは、同じサブネット内のすべてのキャッシュノードに接続できます。

8 インターネットからアクセス可能な Amazon VPC 内の Amazon EC2 インスタンスの場合は、インスタンスに Elastic IP アドレスと呼ばれる静的なパブリックアドレスを割り当てる必要があります。

前提条件

Amazon VPC 内に ElastiCache クラスターを作成するには、Amazon VPC が以下の要件を満たしている必要があります。

- Amazon VPC では、専用ではない Amazon EC2 インスタンスを許可する必要があります。ハードウェア専用インスタンスのテナンシー用に設定された Amazon VPC では ElastiCache を使用できません。

- Amazon VPC 用にキャッシュサブネットグループを定義する必要があります。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内で VPC エンドポイントまたはキャッシュノードに関連付けるサブネットおよび IP アドレスを選択します。
- 各サブネットの CIDR ブロックは、メンテナンス作業で使用する予備の IP アドレスを ElastiCache に提供するのに十分な大きさが必要です。

ルーティングとセキュリティ

Amazon VPC でルーティングを設定して、トラフィックの送信先 (インターネットゲートウェイ、仮想プライベートゲートウェイなど) を制御できます。インターネットゲートウェイの場合、Amazon VPC は、同じ Amazon VPC で実行されているのではない他の AWS リソースに直接アクセスできます。お客様の組織のローカルネットワークに接続された仮想プライベートゲートウェイのみを選択した場合、VPN 経由でインターネット宛てのトラフィックをルーティングし、ローカルセキュリティポリシーとファイアウォールを使用して送信を制御できます。この場合、インターネット経由で AWS リソースにアクセスする際に、帯域の追加料金が発生します。

Amazon VPC セキュリティグループを使用して、Amazon VPC 内の ElastiCache クラスターと Amazon EC2 インスタンスをセキュリティで保護することができます。セキュリティグループは、サブネットレベルでなくインスタンスレベルでファイアウォールのように動作します。

Note

基礎となる IP アドレスは変わる可能性があるため、キャッシュノードに接続するには DNS 名を使用することを強くお勧めします。

「Amazon VPC ドキュメント」

Amazon VPC に関するドキュメントには、Amazon VPC の作成および使用方法について説明する独自のドキュメントがあります。Amazon VPC ガイドへのリンクについて以下の表にまとめます。

説明	ドキュメント
Amazon VPC の使用を開始する方法	Amazon VPC の開始方法
AWS Management Console を通じて Amazon VPC を使用する方法	Amazon VPC User Guide

説明	ドキュメント
すべての Amazon VPC コマンドの詳細説明	Amazon EC2 コマンドラインリファレンス (Amazon VPC コマンドは、Amazon EC2 リファレンスに記載されています)
Amazon VPC API オペレーション、データタイプ、およびエラーの詳細説明	Amazon EC2 API リファレンス (Amazon VPC API オペレーションは、Amazon EC2 リファレンスに記載されています)
オプションとして IPsec VPN 接続のゲートウェイを設定する必要があるネットワーク管理者向け情報	AWS Site-to-Site VPN とは

Amazon Virtual Private Cloud の詳細については、「[Amazon Virtual Private Cloud](#)」を参照してください。

Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン

Amazon は、Amazon VPC 内のキャッシュにアクセスするための以下のシナリオ ElastiCache をサポートしています。

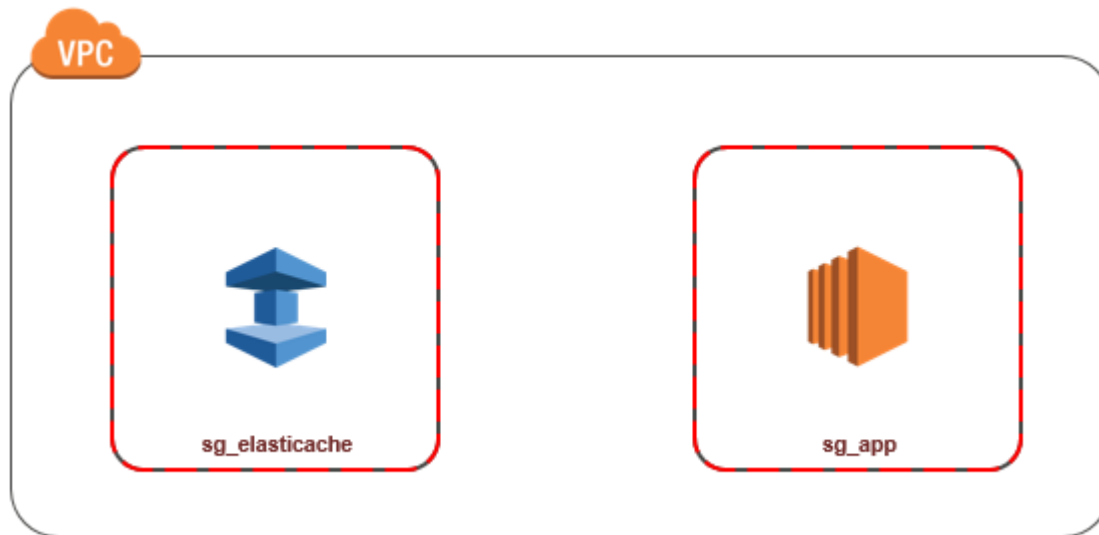
目次

- [ElastiCache キャッシュと Amazon EC2 インスタンスが同じ Amazon VPC にある場合のキャッシュへのアクセス](#)
- [ElastiCache キャッシュと Amazon EC2 インスタンスが異なる Amazon VPCs にある場合のキャッシュへのアクセス](#)
 - [ElastiCache キャッシュと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPCs にある場合のキャッシュへのアクセス](#)
 - [トランジット・ゲートウェイの使用](#)
 - [ElastiCache キャッシュと Amazon EC2 インスタンスが異なるリージョンの異なる Amazon VPCs にある場合のキャッシュへのアクセス](#)
 - [トランジット VPC の使用](#)
- [お客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)
 - [VPN 接続を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)
 - [Direct Connect を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)

ElastiCache キャッシュと Amazon EC2 インスタンスが同じ Amazon VPC にある場合のキャッシュへのアクセス

最も一般的なユースケースは、EC2 インスタンスにデプロイされたアプリケーションが同じ VPC のキャッシュに接続する必要がある場合です。

このシナリオを以下に図表で示します。



同じ VPC 内の EC2 インスタンスとキャッシュ間のアクセスを管理する方法として最も簡単なのは、次の方法です。

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

Redis キャッシュのデフォルトポートは 6379 です。

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

Note

[ローカルゾーン](#) の使用を予定している場合、有効になっていることを確認します。そのローカルゾーンにサブネットグループを作成すると、VPC はそのローカルゾーンに拡張さ

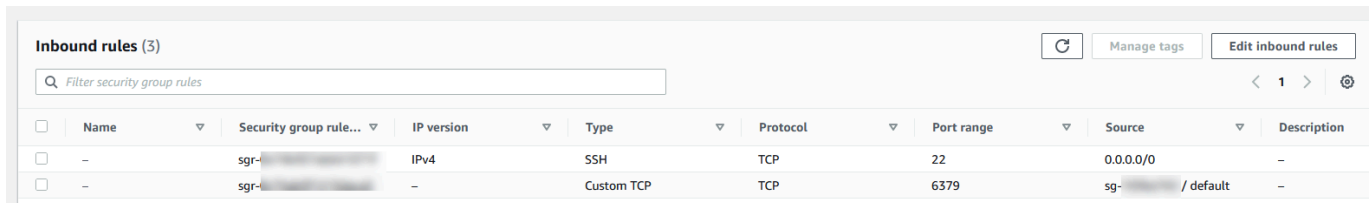
れ、VPC はそのサブネットを他のアベイラビリティーゾーンのサブネットとして扱います。関連するすべてのゲートウェイとルートテーブルが自動的に調整されます。

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

1. AWS マネジメントコンソールにサインインし、<https://console.aws.amazon.com/vpc> で Amazon VPC コンソールを開きます。
2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. キャッシュに使用するセキュリティグループを選択または作成します。インバウンドルールで、インバウンドルールの編集 を選択し、ルールの追加 を選択します。このセキュリティグループは、他のセキュリティグループのメンバーへのアクセスを許可します。
4. Type で Custom TCP Rule を選択します。
 - a. [ポート範囲] には、クラスター作成時に使用したポートを指定します。

Redis キャッシュとレプリケーショングループのデフォルトのポートは 6379 です。

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



<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sg-...	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	sg-...	-	Custom TCP	TCP	6379	sg-... / default	-

ElastiCache キャッシュと Amazon EC2 インスタンスが異なる Amazon VPCs にある場合のキャッシュへのアクセス

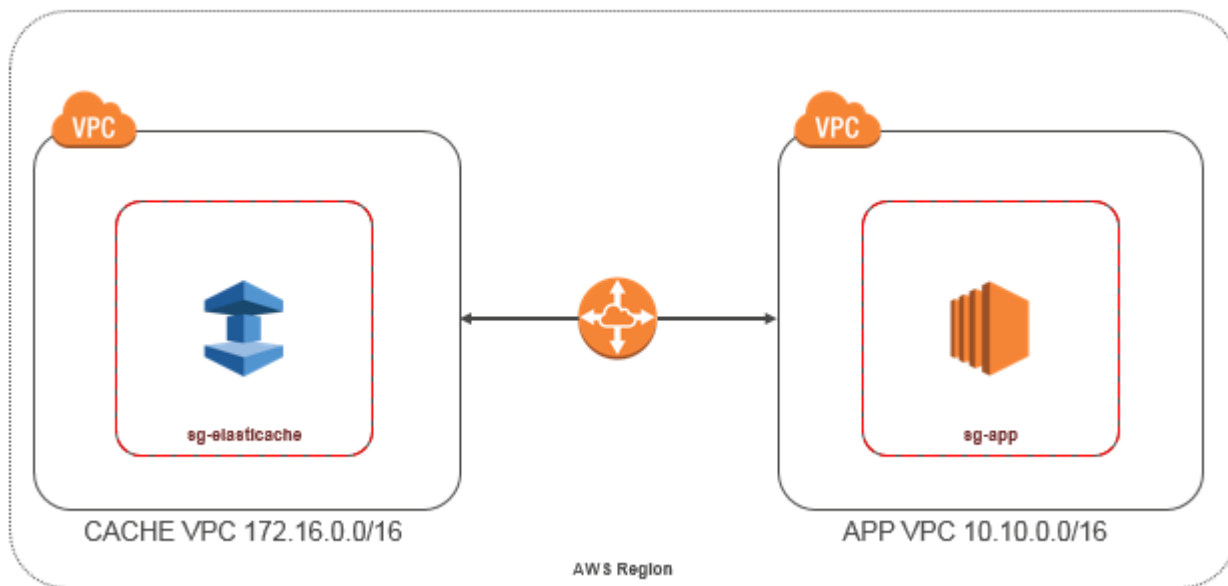
キャッシュがアクセスに使用している EC2 インスタンスとは異なる VPC にある場合、そのキャッシュにアクセスする方法はいくつかあります。キャッシュと EC2 インスタンスが異なる VPC にあるが、同じリージョンにある場合は、VPC ピアリングを使用できる。キャッシュと EC2 インスタンスが異なるリージョンにある場合、リージョン間で VPN 接続を作成できる。

トピック

- [ElastiCache キャッシュと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPCs にある場合のキャッシュへのアクセス](#)
- [ElastiCache キャッシュと Amazon EC2 インスタンスが異なるリージョンの異なる Amazon VPCs にある場合のキャッシュへのアクセス](#)

ElastiCache キャッシュと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPCs にある場合のキャッシュへのアクセス

次の図は、同じリージョンの異なる Amazon VPC での、Amazon VPC ピアリング接続を使用した、Amazon EC2 インスタンスによるキャッシュへのアクセスを示しています。



同じリージョンの異なる Amazon VPC で Amazon EC2 インスタンスによってアクセスされるキャッシュ - VPC ピアリング接続

VPC ピア接続は、プライベート IP アドレスを使用して 2 つの VPC 間でトラフィックをルーティングすることを可能にするネットワーク接続です。どちらの VPC のインスタンスも、同じネットワーク内に存在しているかのように、相互に通信できます。独自の Amazon VPC 間、または単一のリージョン内の別の AWS アカウントの Amazon VPC との間で VPC VPCs ピアリング接続を作成できます。Amazon VPC ピア接続の詳細については、「[VPC ドキュメント](#)」を参照してください。

Note

ElastiCache VPCsの DNS 名解決が失敗することがあります。これを解決するには、両方の VPC を、DNS ホスト名および DNS 解決に対して有効にする必要があります。詳細については、「[VPC ピアリング接続の DNS 解決を有効にする](#)」を参照してください。

ピアリング接続経由で別の Amazon VPC のキャッシュにアクセスするには

1. 2つの VPC に、重複する IP 範囲がないことを確認します。重複する IP 範囲がある場合、それらをピア接続することができません。
2. 2つの VPC をピア接続します。詳細については、「[VPC ピア接続の作成と使用](#)」を参照してください。
3. ルーティングテーブルを更新します。詳細については、「[VPC ピア接続のルートテーブルを更新する](#)」を参照してください

前述の図に示した例のルートテーブルは次のようになります。pcx-a894f1c1 はピア接続であることに注意してください。

Destination	Target	Destination	Target
172.16.0.0/16	local	10.10.0.0/16	local
10.10.0.0/16	pcx-a894f1c1	0.0.0.0/0	igw-bfdcccd8
		172.16.0.0/16	pcx-a894f1c1

VPC ルーティングテーブル

4. ElastiCache キャッシュのセキュリティグループを変更して、ピア接続された VPC のアプリケーション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ます。詳細については、「[ピア VPC セキュリティグループの参照](#)」を参照してください。

ピアリング接続でキャッシュにアクセスすると、追加のデータ転送コストが発生します。

トランジット・ゲートウェイの使用

トランジットゲートウェイを使用すると、同じ AWS リージョンに VPCsと VPN 接続をアタッチし、それらの間でトラフィックをルーティングできます。トランジットゲートウェイは AWS アカウント間で機能し、AWS Resource Access Manager を使用してトランジットゲートウェイを他のア

カウントと共有できます。トランジットゲートウェイを別の AWS アカウントと共有すると、アカウント所有者は自分の VPCs をトランジットゲートウェイにアタッチできます。どちらのアカウントのユーザーも、アタッチメントをいつでも削除できます。

トランジット・ゲートウェイでマルチキャストを有効にしてから、ドメインに関連付ける VPC アタッチメントを介してマルチキャストソースからマルチキャストグループメンバーにマルチキャストトラフィックを送信できるようにする トランジット・ゲートウェイ マルチキャストドメインを作成できます。

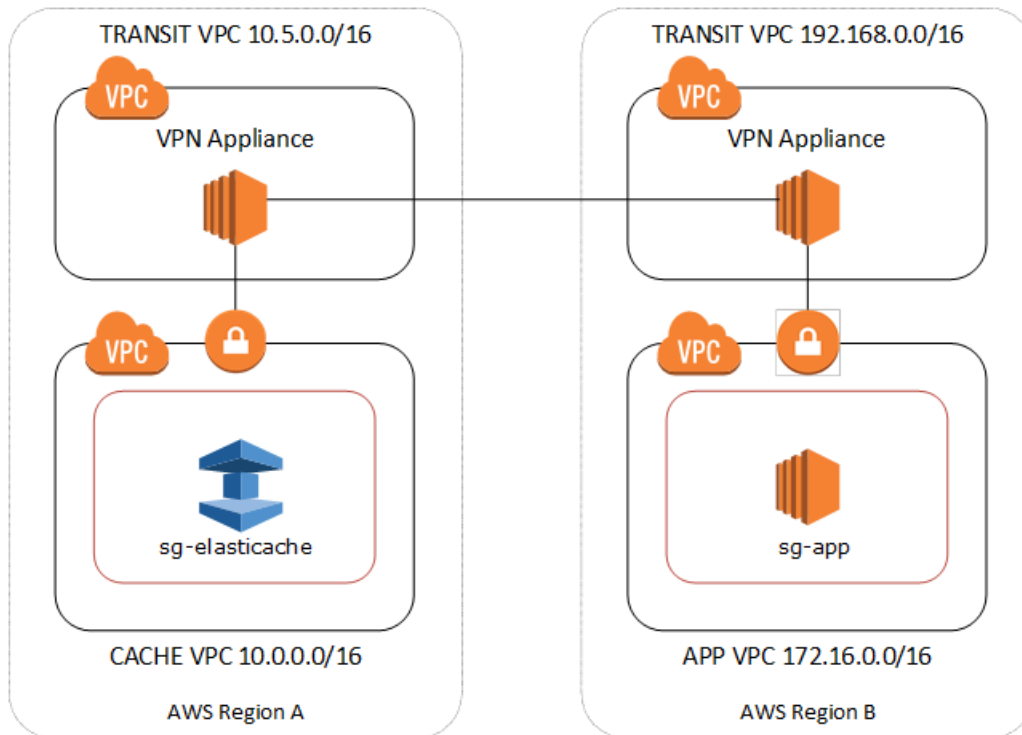
異なるリージョンのトランジットゲートウェイ間にピアリング接続アタッチメントを作成することもできます AWS。これにより、異なるリージョン間でトランジット・ゲートウェイのアタッチメント間でトラフィックをルーティングできます。

詳細については、「[トランジットゲートウェイ](#)」を参照してください。

ElastiCache キャッシュと Amazon EC2 インスタンスが異なるリージョンの異なる Amazon VPCs にある場合のキャッシュへのアクセス

トランジット VPC の使用

VPC ピアリングの代わりに使用する、複数の、地理的に離れた VPC とリモートネットワークを接続する別の一般的な方法は、グローバルなネットワーク中継センターとして機能する中継 VPC の作成です。中継 VPC はネットワーク管理を単純化して、複数の VPC とリモートのネットワークを接続するために必要な接続数を最小限に抑えます。この設計は、コロケーション中継ハブを物理的に設立したり、物理的なネットワーク設備をデプロイしたりするための従来の費用をほとんどかけずに実装できるため、時間と労力を節約し、コストも削減できます。



異なるリージョンの異なる VPC 間での接続

Transit Amazon VPC が確立されると、あるリージョンの「スポーク」VPC にデプロイされたアプリケーションは、別のリージョン内の「スポーク」VPC の ElastiCache キャッシュに接続できます。

別の AWS リージョン内の別の VPC のキャッシュにアクセスするには

1. Transit VPC ソリューションをデプロイします。詳細については、「[AWS Transit Gateway](#)」を参照してください。
2. アプリおよびキャッシュ VPC の VPC ルーティングテーブルを更新し、VGW (仮想プライベートゲートウェイ) および VPN アプライアンスを通じてトラフィックをルーティングします。ボーダーゲートウェイプロトコル (BGP) を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. ElastiCache キャッシュのセキュリティグループを変更して、アプリケーションインスタンスの IP 範囲からのインバウンド接続を許可します。このシナリオでは、アプリケーションサーバーセキュリティグループを参照することはできません。

リージョン間でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、リージョン間のデータ転送コストが追加で発生します。

お客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

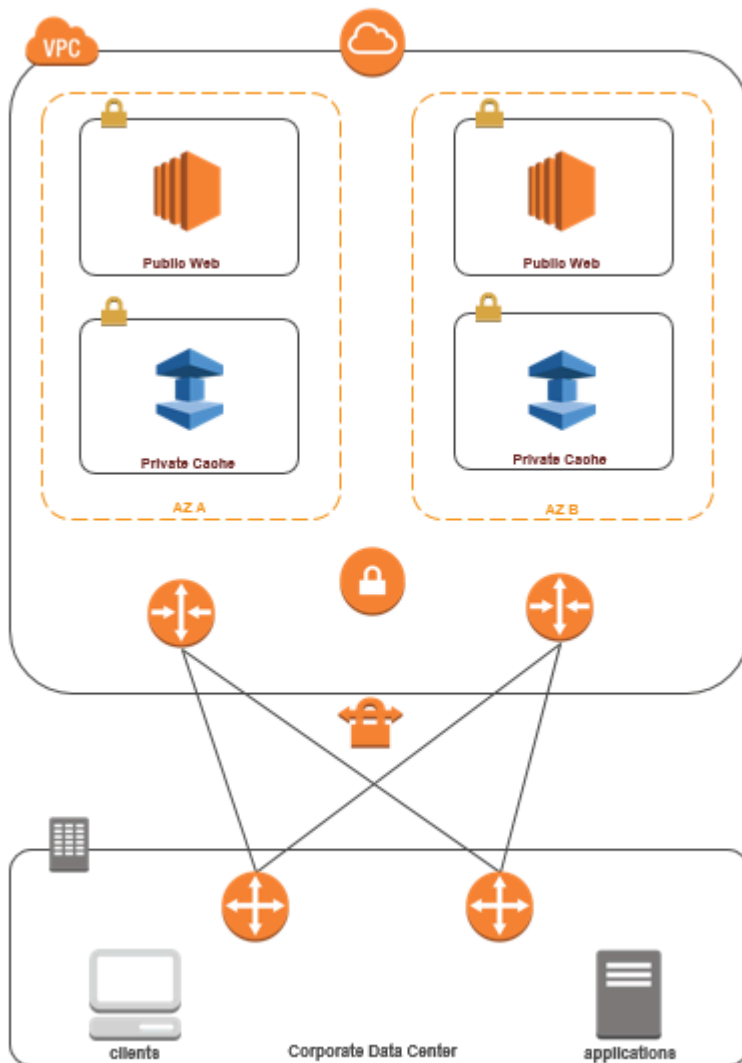
もう 1 つのシナリオは、顧客のデータセンター内のクライアントまたはアプリケーションが VPC 内の ElastiCache キャッシュにアクセスする必要があるハイブリッドアーキテクチャです。このシナリオは、顧客の VPC とデータセンター間で VPN または Direct Connect による接続がある場合にサポートされます。

トピック

- [VPN 接続を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)
- [Direct Connect を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする](#)

VPN 接続を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

次の図は、VPN 接続を使用して企業ネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスする方法を示しています。



VPN 経由でデータセンター ElastiCache から に接続する

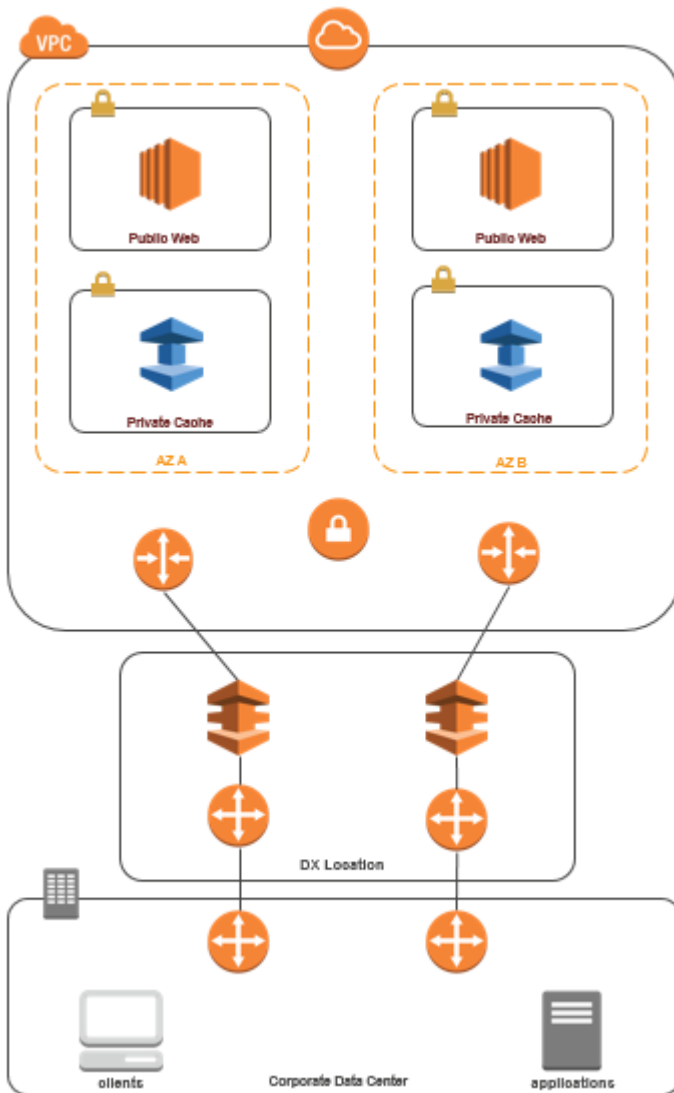
VPN 接続経由でオンプレミスアプリケーションから VPC のキャッシュにアクセスするには

1. VPC にハードウェア仮想プライベートゲートウェイを追加して、VPN 接続を確立します。詳細については、「[VPC へのハードウェア仮想プライベートゲートウェイの追加](#)」を参照してください。
2. ElastiCache キャッシュがデプロイされているサブネットの VPC ルーティングテーブルを更新して、オンプレミスアプリケーションサーバーからのトラフィックを許可します。BGP を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. ElastiCache キャッシュのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

VPN 接続経由でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送コストが発生します。

Direct Connect を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

次の図は、Direct Connect を使用して企業ネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスする方法を示しています。



Direct Connect 経由でデータセンター ElastiCache から に接続する

Direct Connect を使用してネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスするには

1. Direct Connect 接続を確立します。詳細については、[AWS 「Direct Connect の開始方法」](#)を参照してください。
2. ElastiCache キャッシュのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

DX 接続経由でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送料金が発生する場合があります。

Virtual Private Cloud (VPC) の作成

この例では、各アベイラビリティゾーンのパブリックサブネットを持つ Amazon VPC を作成します。

Amazon VPC の作成 (コンソール)

1. AWS マネジメントコンソールにサインインして Amazon VPC コンソール (<https://console.aws.amazon.com/vpc/>) を開きます。
2. VPC ダッシュボードで、Create VPC (VPC の作成) を選択します。
3. Resources to create (作成するリソース) で、VPC only (VPC など) を選択します。
4. Number of Availability Zones (AZs) (アベイラビリティゾーンの数 (AZ)) で、サブネットを起動するアベイラビリティゾーンの数を選択します。
5. Number of public subnets (パブリックサブネットの数) で、VPC に追加するパブリックサブネットの数を選択します。
6. Number of private subnets (プライベートサブネットの数) で、VPC に追加するプライベートサブネットの数を選択します。

Tip

サブネットの識別子と、どちらがパブリックで、どちらがプライベートであるかを書き留めておきます。この情報は、後でクラスターを起動し、Amazon VPC に Amazon EC2 インスタンスを追加するときに必要になります。

7. Amazon VPC セキュリティグループを作成します。キャッシュクラスターと Amazon EC2 インスタンスでは、このグループを使用します。
 - a. Amazon VPC Management Console のナビゲーションペインで、[Security Group (セキュリティグループ)] を選択します。
 - b. [Create Security Group (セキュリティグループの作成)] を選択します。
 - c. 対応するボックスにセキュリティグループの名前と説明を入力します。[VPC] ボックスで Amazon VPC の ID を選択します。

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)
my-vpc-security-group
Name cannot be edited after creation.

Description [Info](#)
my vpc security group

VPC [Info](#)
vpc-862574fc

Inbound rules [Info](#)

This security group has no inbound rules.

[Add rule](#)

Outbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info
All traffic	All	All	Custom	

0.0.0.0/0 [X](#) [Delete](#)

[Add rule](#)

- d. すべての設定が正しいことを確認したら、Yes, Create を選択します。
8. セキュリティグループのネットワーク Ingress ルールを定義します。このルールは、Secure Shell (SSH) を使用して Amazon EC2 インスタンスに接続することを許可します。
- a. ナビゲーションリストで [Security Groups] を選択します。
 - b. リストで対象となるセキュリティグループを探して選択します。
 - c. Security Group の下で、Inbound タブを選択します。Create a new rule ボックスで、SSH を選択し、Add Rule を選択します。
 - d. 新しいインバウンドルールに次の値を設定して、HTTP へのアクセスを許可します。
 - Type: HTTP
 - ソース: 0.0.0.0/0

Apply Rule Changes を選択します。

これで、キャッシュサブネットグループを作成して Amazon VPC でキャッシュクラスターを起動する準備が整いました。

- [サブネットグループの作成](#)
- [Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#).

Amazon VPC で実行されるキャッシュへの接続

この例では、Amazon VPC で Amazon EC2 インスタンスを起動する方法を示します。次に、このインスタンスにログインし、その Amazon VPC で実行中の ElastiCache キャッシュにアクセスできます。

Amazon VPC で実行されるキャッシュへの接続 (コンソール)

この例では、Amazon VPC で Amazon EC2 インスタンスを作成します。この Amazon EC2 インスタンスを使用して、Amazon VPC で実行中のキャッシュノードに接続できます。

Note

Amazon EC2 の使用に関する詳細は、[Amazon EC2 ドキュメント](#)の「[Amazon EC2 入門ガイド](#)」を参照してください。

Amazon EC2 コンソールを使用して Amazon VPC で Amazon EC2 インスタンスを作成するには

1. AWS Management Console にサインインし、Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2/>) を開きます。
2. コンソールで、[Launch Instance (インスタンスの起動)] を選択し、以下の手順を実行します。
3. [Choose an Amazon Machine Image (AMI) (Amazon マシンイメージ (AMI) の選択)] ページで、64 ビット Amazon Linux AMI を選択し、[選択] を選択します。
4. [インスタンスタイプの選択] ページで、[3.] を選択します。[インスタンスの設定]。
5. [インスタンスの詳細の設定] ページで以下の項目を選択します。
 - a. [ネットワーク] リストで、Amazon VPC を選択します。
 - b. [サブネット] リストで、パブリックサブネットを選択します。

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage pricing, assign an access management role to the instance, and more.

Number of instances ⓘ

Purchasing option ⓘ Request Spot Instances

Network ⓘ

Subnet ⓘ
250 IP Addresses available

Public IP ⓘ Automatically assign a public IP address to your instances

すべての設定が正しいことを確認したら、[4.] を選択します。[Add Storage] (ストレージの追加)。

6. [ストレージの追加] ページで、[5.] を選択します。[Tag Instance] (インスタンスのタグ付け)。
7. [インスタンスのタグ付け] ページで、Amazon EC2 インスタンスの名前を入力し、[6.] を選択します。[Configure Security Group] (セキュリティグループの設定)。
8. [セキュリティグループの設定] ページで [Select an existing security group (既存のセキュリティグループの選択)] を選択します。セキュリティグループの詳細については、「[Linux インスタンス用の Amazon EC2 セキュリティグループ](#)」を参照してください。

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP. You can create a new security group or select from an existing one below. [Learn more about Amazon EC2 security groups.](#)

Assign a security group: Create a new security group
 Select an existing security group

Security Group ID	Name	Description
<input type="checkbox"/> sg-1a3d2178	default	default VPC security group
<input checked="" type="checkbox"/> sg-f13d2193	my-vpc-security-group	Testing

Amazon VPC セキュリティグループの名前を選択し、[Review and Launch (確認および起動)] を選択します。

9. [Review Instance and Launch (インスタンスの確認および起動)] ページで、[起動] を選択します。
10. [Select an existing key pair or create a new key pair (既存のキーペアを選択するか新しいキーペアを作成する)] ウィンドウで、このインスタンスで使用するキーペアを指定します。

Note

キーペアの管理の詳細については、「[Amazon EC2 入門ガイド](#)」を参照してください。

11. Amazon EC2 インスタンスを起動する準備ができたなら、[起動] を選択します。

これで、先ほど作成した Amazon EC2 インスタンスに Elastic IP アドレスを割り当てることができま
す。この IP アドレスは、Amazon EC2 インスタンスに接続するときに使用する必要があります。

Elastic IP アドレスを割り当てするには (コンソール)

1. Amazon VPC コンソール (<https://console.aws.amazon.com/vpc/>) を開きます。
2. ナビゲーションリストで [Elastic IP] を選択します。
3. [Elastic IP アドレスの割り当て] を選択します。
4. [Allocate Elastic IP address (Elastic IP アドレスを割り当てる)] ダイアログボックスで、デフォ
ルトの [ネットワークボーダーグループ] を受け入れ、[割り当て] を選択します。
5. リストから割り当てた Elastic IP アドレスを選択し、[アドレスの関連付け] を選択します。
6. [アドレスの関連付け] ダイアログボックスの [インスタンス] ボックスで、起動した Amazon
EC2 インスタンスの ID を選択します。

[プライベート IP アドレス] ボックスで、プライベート IP アドレスを取得するボックスを選択
し、[関連付け] を選択します。

これで、SSH を使用して、作成した Elastic IP アドレスを使用して Amazon EC2 インスタンス
に接続できるようになりました。

Amazon EC2 インスタンスに接続するには

- コマンドウィンドウを開きます。コマンドプロンプトで、次のコマンドを実行しま
す。mykeypair.pem をご使用のキーペアファイルの名前に、54.207.55.251 をご使用の Elastic
IP アドレスに置き換えます。

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

⚠ Important

まだ Amazon EC2 インスタンスからログアウトしないでください。

これで、ElastiCache クラスターを操作する準備ができました。まだ telnet ユーティリティをインストールしていない場合、これを行うには、このユーティリティをインストールする必要があります。

telnet をインストールし、キャッシュクラスター (AWS CLI) を操作するには

1. コマンドウィンドウを開きます。コマンドプロンプトで、次のコマンドを発行します。確認のプロンプトが表示されたら、「y」を入力します。

```
sudo yum install telnet
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                | 63 kB      00:00

...(output omitted)...

Complete!
```

2. telnet を使用して、ポート 6379 でキャッシュノードのエンドポイントに接続します。以下に示すホスト名をキャッシュノードのホスト名に置き換えます。

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

これで、キャッシュエンジンに接続され、コマンドを実行できます。この例では、キャッシュにデータ項目を追加し、その後すぐにそれを取得します。最後に、キャッシュノードから切断します。

キーと値を保存するには、次の 2 行を入力します。

```
set mykey myvalue
```

キャッシュエンジンは以下のように応答します。

```
OK
```

[mykey] の値を取得するには、次のように入力します。

```
get mykey
```

キャッシュエンジンから切断するには、次のように入力します。

```
quit
```

3. ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) に移動して、キャッシュクラスターのいずれかのノードのエンドポイントを取得します。詳細については、Redis の「[接続エンドポイントの検索](#)」を参照してください。
4. telnet を使用して、ポート 6379 でキャッシュノードのエンドポイントに接続します。以下に示すホスト名をキャッシュノードのホスト名に置き換えます。

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

これで、キャッシュエンジンに接続され、コマンドを実行できます。この例では、キャッシュにデータ項目を追加し、その後すぐにそれを取得します。最後に、キャッシュノードから切断します。

キーと値を保存するには、次のように入力します。

```
set mykey myvalue
```

キャッシュエンジンは以下のように応答します。

```
OK
```

[mykey] の値を取得するには、次のように入力します。

```
get mykey
```

キャッシュエンジンは以下のように応答します。

```
get mykey  
myvalue
```

キャッシュエンジンから切断するには、次のように入力します。

```
quit
```

Important

AWS アカウントの追加料金が発生しないように、これらの例を試した後で、今後使用しない AWS リソースはすべて削除してください。

Amazon ElastiCache API とインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、VPC と Amazon ElastiCache API エンドポイント間にプライベート接続を確立できます。インターフェイスエンドポイントは [AWS PrivateLink](#) を使用します。AWS PrivateLink を使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで、Amazon ElastiCache API オペレーションにプライベートにアクセスできます。

VPC のインスタンスは、パブリック IP アドレスがなくても Amazon ElastiCache API エンドポイントと通信できます。また、ElastiCache API オペレーションの使用にも、パブリック IP アドレスを必要としません。VPC と Amazon ElastiCache 間のトラフィックは、Amazon ネットワークを離れません。各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。Elastic Network Interface の詳細については、Amazon EC2 ユーザーガイドの「[Elastic Network Interface](#)」を参照してください。

- VPC エンドポイントの詳細については、Amazon VPC ユーザーガイドの「[インターフェイス VPC エンドポイント \(AWS PrivateLink\)](#)」を参照してください。

- ElastiCache API オペレーションの詳細については、「[ElastiCache API オペレーション](#)」を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントの[プライベート DNS](#) ホスト名を有効にすると、デフォルトの ElastiCache エンドポイント (<https://elasticache.Region.amazonaws.com>) はお客様の VPC エンドポイントに解決されます。プライベート DNS ホスト名を有効にしない場合は、Amazon VPC が以下の形式で利用できる DNS エンドポイント名を提供します。

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

詳細については、Amazon VPC ユーザーガイドの「AWS インターフェイス VPC エンドポイント ([PrivateLink](#))」をご参照ください。ElastiCache は、VPC 内のすべての [API アクション](#) への呼び出しをサポートしています。

Note

プライベート DNS ホスト名は、VPC 内の 1 つの VPC エンドポイントに対してのみ有効にできます。追加の VPC エンドポイントを作成する場合は、プライベート DNS ホスト名を無効にする必要があります。

VPC エンドポイントに関する考慮事項

Amazon ElastiCache API エンドポイントのインターフェイス VPC エンドポイントを設定する前に、「Amazon VPC ユーザーガイド」の「[インターフェイスエンドポイントのプロパティと制限](#)」を確認してください。Amazon ElastiCache リソースの管理に関連するすべての ElastiCache 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 ElastiCache 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 ElastiCache API オペレーションへのフルアクセスが許可されます。詳細については、Amazon VPC ユーザーガイドの[VPC エンドポイントによるサービスのアクセスコントロール](#)を参照してください。

ElastiCache API 用のインターフェイス VPC エンドポイントの作成

Amazon ElastiCache API 用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS CLI で作成できます。詳細については、Amazon VPC ユーザーガイドの[インターフェイスエンドポイントの作成](#)を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にできます。その場合、デフォルトの Amazon ElastiCache エンドポイント (<https://elasticache.Region.amazonaws.com>) は、お客様の VPC エンドポイントに解決されます。中国 (北京) および 中国 (寧夏) AWS リージョンの場合、北京に elasticache.cn-north-1.amazonaws.com.cn、寧夏に elasticache.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 VPC ユーザーガイド」の「[インターフェイスエンドポイントを介したサービスへのアクセス](#)」を参照してください。

Amazon ElastiCache API 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに ElastiCache API へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。本ポリシーでは、以下を規定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「[VPC エンドポイントでサービスへのアクセスを制御する](#)」を参照してください。

Example ElastiCache API アクションの VPC エンドポイントポリシー

ElastiCache API のエンドポイントポリシーの例を次に示します。このポリシーは、エンドポイントにアタッチされると、すべてのリソースのすべてのプリンシパルに対して、登録されている ElastiCache API アクションへのアクセスを許可します。

```
{
  "Statement": [{
    "Principal": "*",
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:ModifyCacheCluster",
      "elasticache:CreateSnapshot"
    ],
    "Resource": "*"
  }]
}
```

Example 指定した AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、AWS アカウント **123456789012** がエンドポイントを使用するリソースへのすべてのアクセスを拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
  "Statement": [{
    "Action": "*",
    "Effect": "Allow",
    "Resource": "*",
    "Principal": "*"
  },
  {
    "Action": "*",
    "Effect": "Deny",
    "Resource": "*",
    "Principal": {
      "AWS": [
        "123456789012"
      ]
    }
  }
]
```

サブネットおよびサブネットグループ

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

Amazon VPC で独自設計型クラスターを作成する場合、サブネットグループを使用する必要があります。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。

ElastiCache はデフォルトの IPv4 サブネットグループを提供しています。または、新しいサブネットグループを作成することもできます。IPv6 の場合は、IPv6 CIDR ブロックを使用するサブネットグループを作成する必要があります。デュアルスタックを選択した場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

ElastiCache サーバーレスはサブネットグループリソースを使用せず、代わりに作成時にサブネットのリストを直接取得します。

このセクションでは、サブネットおよびサブネットグループを作成し活用して、ElastiCache リソースへのアクセスを管理する方法を扱います。

Amazon VPC 環境でのサブネットグループの使用方法の詳細については、「[クラスターまたはレプリケーショングループへのアクセス](#)」を参照してください。

トピック

- [サブネットグループの作成](#)
- [キャッシュにサブネットグループを割り当てる](#)
- [サブネットグループの変更](#)
- [サブネットグループの削除](#)

サブネットグループの作成

キャッシュサブネットグループは、VPC 内でとして指定できるサブネットの集合です。VPC でキャッシュを起動する場合は、キャッシュサブネットグループを選択する必要があります。次に、ElastiCache ではそのキャッシュサブネットグループを使用して、サブネット内の IP アドレスをキャッシュ内の各キャッシュノードに割り当てます。

新しいサブネットグループを作成する場合は、使用可能な IP アドレス数に注意してください。サブネットの空き IP アドレス数が非常に少ない場合は、クラスターに追加できるノード数が制約される可能性があります。この問題を解決するために、クラスターのアベイラビリティゾーンで十分な数の IP アドレスを使用できるように、サブネットグループに 1 つ以上のサブネットを割り当てることができます。その後で、クラスターにノードを追加できます。

ネットワークタイプとして IPv4 を選択した場合は、デフォルトのサブネットグループが利用でき、新しいサブネットグループを作成することもできます。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。デュアルスタックまたは IPV6 を選択すると、デュアルスタックまたは IPV6 サブネットを作成するように指示されます。ネットワークタイプの詳細については、「[ネットワークタイプ](#)」を参照してください。詳細については、「[VPC にサブネットを作成する](#)」を参照してください。

以下の手順では、mysubnetgroup (コンソール)、AWS CLI、および ElastiCache API というサブネットグループを作成する方法を示します。

サブネットグループの作成 (コンソール)

次の手順では、サブネットグループ (コンソール) を作成する方法を示します。

サブネットグループ (コンソール) を作成するには

1. AWS マネジメントコンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションリストで [サブネットグループ] を選択します。
3. [サブネットグループの作成] を選択します。
4. [サブネットグループを作成] ウィザードで、次の操作を行います。すべての設定が正しいことを確認したら、作成 (作成) を選択します。
 - a. Name ボックスにサブネットグループの名前を入力します。
 - b. Description ボックスにサブネットグループの説明を入力します。
 - c. [VPC ID] ボックスで Amazon VPC を選択します。

- d. デフォルトでは、すべてのサブネットが選択されています。[選択済みのサブネット]パネルで [管理] をクリックし、プライベートサブネットのアベイラビリティゾーンまたは [ローカルゾーン](#) と ID を選択し、[選択] をクリックします。
5. 表示された確認メッセージで、Close を選択します。

ElastiCache コンソールの [サブネットグループ] のリストに新しいサブネットグループが表示されます。ウィンドウの下部で、サブネットグループを選択して、ウィンドウの下部で詳細 (このグループに関連付けられているすべてのサブネットなど) を確認します。

サブネットグループの作成 (AWS CLI)

コマンドプロンプトで、`create-cache-subnet-group` コマンドを使用してサブネットグループを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-cache-subnet-group \  
  --cache-subnet-group-name mysubnetgroup \  
  --cache-subnet-group-description "Testing" \  
  --subnet-ids subnet-53df9c3a
```

Windows の場合:

```
aws elasticache create-cache-subnet-group ^  
  --cache-subnet-group-name mysubnetgroup ^  
  --cache-subnet-group-description "Testing" ^  
  --subnet-ids subnet-53df9c3a
```

このコマンドでは、次のような出力が生成されます。

```
{  
  "CacheSubnetGroup": {  
    "VpcId": "vpc-37c3cd17",  
    "CacheSubnetGroupDescription": "Testing",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-53df9c3a",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2a"  
        }  
      }  
    ]  
  }  
}
```

```
    }  
  ],  
  "CacheSubnetGroupName": "mysubnetgroup"  
}  
}
```

詳細については、AWS CLI のトピック「[create-cache-subnet-group](#)」を参照してください。

キャッシュにサブネットグループを割り当てる

サブネットグループを作成したら、Amazon VPC でキャッシュを起動できます。詳細については、以下を参照してください。

- [スタンドアロン Redis クラスター] – 単一ノード Redis クラスターを起動するには、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」を参照してください。ステップ 7。a [Advanced Redis Settings] で、VPC サブネットグループを選択します。
- Redis (クラスターモードが無効) レプリケーショングループ – VPC で Redis (クラスターモードが無効) レプリケーショングループを起動するには、「[Redis \(クラスターモードが無効\) レプリケーショングループを最初から作成する](#)」を参照してください。ステップ 7。b [Advanced Redis Settings] で、VPC サブネットグループを選択します。
- [Redis (クラスターモードが有効) レプリケーショングループ] – 「[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#)」。ステップ 6。i [Advanced Redis Settings] で、VPC サブネットグループを選択します。

サブネットグループの変更

サブネットグループの説明を変更することや、サブネットグループに関連付けられたサブネット ID のリストを変更することができます。キャッシュが現在サブネットを使用している場合、サブネットグループからそのサブネット ID を削除することはできません。

次の手順では、サブネットグループを変更する方法を示します。

サブネットグループの変更 (コンソール)

サブネットグループを変更するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) 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。
3. サブネットグループのリストで、変更するグループのラジオボタンを選択し、[変更] をクリックします。
4. [選択済みサブネット] パネルで [管理] を選択します。
5. 選択したサブネットに変更を加えて、[選択] をクリックします。
6. [保存] をクリックして変更を保存します。

サブネットグループの変更 (AWS CLI)

コマンドプロンプトで、`modify-cache-subnet-group` コマンドを使用してサブネットグループを変更します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-subnet-group \  
  --cache-subnet-group-name mysubnetgroup \  
  --cache-subnet-group-description "New description" \  
  --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Windows の場合:

```
aws elasticache modify-cache-subnet-group ^  
  --cache-subnet-group-name mysubnetgroup ^  
  --cache-subnet-group-description "New description" ^  
  --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

このコマンドでは、次のような出力が生成されます。

```
{
  "CacheSubnetGroup": {
    "VpcId": "vpc-73cd3c17",
    "CacheSubnetGroupDescription": "New description",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-42dcf93a",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      },
      {
        "SubnetIdentifier": "subnet-48fc12a9",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      }
    ],
    "CacheSubnetGroupName": "mysubnetgroup"
  }
}
```

詳細については、AWS CLI のトピック「[modify-cache-subnet-group](#)」を参照してください。

サブネットグループの削除

サブネットグループが必要ではなくなったと判断した場合、サブネットグループを削除できます。サブネットグループがキャッシュで現在使用されている場合、サブネットグループを削除できません。

次の手順では、サブネットグループを削除する方法を示します。

サブネットグループの削除 (コンソール)

サブネットグループを削除するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) 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。
3. サブネットグループのリストで、削除する項目を選択して [Delete] を選択します。
4. この操作を確定するように求められたら、テキスト入力フィールドにサブネットグループの名前を入力し、[削除] を選択します。

サブネットグループの削除 (AWS CLI)

AWS CLI を使用して、以下のパラメーターでコマンド `delete-cache-subnet-group` を呼び出します。

- `--cache-subnet-group-name mysubnetgroup`

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-subnet-group \  
  --cache-subnet-group-name mysubnetgroup
```

Windows の場合:

```
aws elasticache delete-cache-subnet-group ^  
  --cache-subnet-group-name mysubnetgroup
```

このコマンドでは何も出力されません。

詳細については、AWS CLI のトピック「[delete-cache-subnet-group](#)」を参照してください。

Amazon の Identity and Access Management ElastiCache

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つです。IAM 管理者は、誰を認証 (サインイン) し、誰に ElastiCache リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

- [対象者](#)
- [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 と IAM ElastiCache の連携方法](#)
- [Amazon ElastiCache のアイデンティティベースのポリシーの例](#)
- [Amazon ElastiCache アイデンティティとアクセスのトラブルシューティング](#)
- [アクセスコントロール](#)
- [ElastiCache リソースに対するアクセス許可の管理の概要](#)

対象者

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります ElastiCache。

サービスユーザー – ElastiCache サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの ElastiCache 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておく、管理者に適切な許可をリクエストするうえで役立ちます。の機能にアクセスできない場合は、ElastiCache 「」を参照してください [Amazon ElastiCache アイデンティティとアクセスのトラブルシューティング](#)。

サービス管理者 – 社内の ElastiCache リソースを担当している場合は、通常、 へのフルアクセスがあります ElastiCache。サービスユーザーがどの ElastiCache 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を で使用する方法の詳細については ElastiCache、 「」を参照してください [Amazon と IAM ElastiCache の連携方法](#)。

IAM 管理者 - IAM 管理者は、へのアクセス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 ElastiCache。IAM で使用できる ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon ElastiCache のアイデンティティベースのポリシーの例](#)。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーテッド ID AWS として にサインインできます。AWS IAM Identity Center (IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーテッド ID の例です。フェデレーテッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることとなります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「ユーザーガイド」の [「へのサインイン AWS アカウント」](#)方法AWS サインイン」を参照してください。

AWS プログラムで にアクセスする場合、は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM [ユーザーガイドの API AWS リクエスト](#)の署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、AWS では、アカウントのセキュリティを強化するために多要素認証 (MFA) を使用することをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の [「多要素認証」](#) および「IAM ユーザーガイド」の [「AWSでの多要素認証 \(MFA\) の使用」](#)を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの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、一時的な認証情報を使用してにアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービスします。

フェデレーテッド 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のユーザー、または ID ソースを通じ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。フェデレーテッド ID がにアクセスすると 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ことも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[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 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「[長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を参照してください。

[IAM グループ](#)は、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー

ザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「[IAM ユーザー \(ロールではなく\) の作成が適している場合](#)」を参照してください。

IAM ロール

[IAM ロール](#)は、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、[で IAM ロール](#)を一時的に引き受けることができます。ロールを引き受けるには、または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「[IAM ロールの使用](#)」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス - フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「[Creating a role for a third-party Identity Provider](#)」(サードパーティーアイデンティティプロバイダー向けロールの作成)を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可セット](#)」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、IAM ユーザーガイドの「[IAM でのクロスアカウントリソースアクセス](#)」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する](#)」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「[\(IAM ユーザーではなく\) IAM ロールをいつ作成したら良いのか?](#)」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、AWS ID またはリソースにアタッチします。ポリシーは 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 AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースに必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、AWS Management Console、AWS CLI または AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーの作成](#)」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「[マネージドポリシーとインラインポリシーの比較](#)」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、[プリンシパルを指定する](#)必要があります。プ

リンシパルには、アカウント、ユーザー、ロール、フェデレーテッドユーザー、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、AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「[アクセスコントロールリスト \(ACL\) の概要](#)」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- **アクセス許可の境界** - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「[IAM エンティティのアクセス許可の境界](#)」を参照してください。
- **サービスコントロールポリシー (SCPs)** - SCPs は、の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。AWS Organizations は、AWS アカウント ビジネスが所有する複数のをグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「[SCP の仕組み](#)」を参照してください。
- **セッションポリシー** - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポ

リシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「[セッションポリシー](#)」を参照してください。

複数のポリシータイプ

1つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうかが 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 と IAM ElastiCache の連携方法

IAM を使用してへのアクセスを管理する前に ElastiCache、で利用できる IAM 機能について学びます ElastiCache。

Amazon で使用できる IAM の機能 ElastiCache

IAM 機能	ElastiCache サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	Yes
ポリシー条件キー	Yes
ACL	はい
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	Yes
プリンシパル権限	Yes
サービスロール	あり

IAM 機能	ElastiCache サポート
サービスリンクロール	はい

ElastiCache 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を把握するには、「IAM ユーザーガイド」の[AWS 「IAM と連携する のサービス」](#)を参照してください。

のアイデンティティベースのポリシー ElastiCache

アイデンティティベースポリシーをサポートする **Yes**

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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のポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーの作成](#)」を参照してください。

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。

ElastiCache のアイデンティティベースのポリシーの例

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon ElastiCache のアイデンティティベースのポリシーの例](#)。

ElastiCache 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 **No**

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 がある。

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 がある場合 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。

のポリシーアクション ElastiCache

ポリシーアクションに対するサポート	はい
-------------------	----

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの Action 要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

ElastiCache アクションのリストを確認するには、「サービス認証リファレンス」の[「Amazon で定義されるアクション ElastiCache」](#)を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス ElastiCache を使用します。

```
elasticache
```

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": [  
  "elasticache:action1",  
  "elasticache:action2"  
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "elasticache:Describe*"
```

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon ElastiCache のアイデンティティベースのポリシーの例](#)。

のポリシーリソース ElastiCache

ポリシーリソースに対するサポート はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、[Amazon リソースネーム \(ARN\)](#) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

ElastiCache リソースタイプとその ARNs」の「[Amazon で定義されるリソース ElastiCache](#)」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「[Amazon で定義されるアクション ElastiCache](#)」を参照してください。

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してください。[Amazon ElastiCache のアイデンティティベースのポリシーの例](#)。

ElastiCache 向け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 はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの [条件演算子](#) を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定するか、1 つの Condition 要素に複数のキーを指定すると、AWS は AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「[IAM ポリシーの要素: 変数およびタグ](#)」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」の[AWS 「グローバル条件コンテキストキー」](#)を参照してください。

ElastiCache 条件キーのリストを確認するには、「サービス認証リファレンス」の「[Amazon の条件キー ElastiCache](#)」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 で定義されるアクション ElastiCache](#)」を参照してください。

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してください。[Amazon ElastiCache のアイデンティティベースのポリシーの例](#)。

のアクセスコントロールリスト (ACLs) ElastiCache

ACL のサポート	はい
-----------	----

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コントロールします。ACL はリソーススペ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。

での属性ベースのアクセスコントロール (ABAC) ElastiCache

ABAC のサポート (ポリシー内のタグ)	はい
-----------------------	----

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アクセス許可戦略です。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はいです。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 とは?](#)」を参照してください。ABAC をセットアップするステップ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制御 \(ABAC\) を使用する](#)」を参照してください。

での一時的な認証情報の使用 ElastiCache

一時的な認証情報のサポート	はい
---------------	----

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などの詳細については、IAM ユーザーガイドの[AWS のサービス「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 方法でサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用してアクセスすると、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「[ロールへの切り替え \(コンソール\)](#)」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、AWS recommends にアクセスできます AWS。この際、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「[IAM の一時的セキュリティ認証情報](#)」を参照してください。

ElastiCache のクロスサービスプリンシパル権限

フォワードアクセスセッション (FAS) をサポート

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、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のリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービスまたはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。

のサービスロール ElastiCache

サービスロールに対するサポート **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

では、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。

Warning

サービスロールのアクセス許可を変更すると、ElastiCache 機能が破損する可能性があります。が指示する場合以外 ElastiCache は、サービスロールを編集しないでください。

のサービスにリンクされたロール ElastiCache

サービスリンクロールのサポート	はい
-----------------	----

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権限を表示できますが、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。表の中から、Service-linked role (サービスにリンクされたロール) 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、はい リンクを選択します。

Amazon ElastiCache のアイデンティティベースのポリシーの例

デフォルトで、ユーザーとロールには ElastiCache リソースを作成または変更する許可がありません。また、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースに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、管理者がロールに IAM ポリシーを追加すると、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「[IAM ポリシーの作成](#)」を参照してください。

ElastiCache 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、「サービス認証リファレンス」の「[Amazon ElastiCache 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

- [ポリシーのベストプラクティス](#)
- [ElastiCache コンソールの使用](#)
- [自分の権限の表示をユーザーに許可する](#)

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰が ElastiCache 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 マネージドポリシーを使用して開始し、最小特権の権限に移行する - ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、権限をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」または「[AWS ジョブ機能の管理ポリシー](#)」を参照してください。
- 最小特権を適用する - IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。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、最小特権権限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 でのポリシーと権限](#)」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS CloudFormation などの特定の AWS のサービスを介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、「IAM ユーザーガイド」の「[IAM JSON ポリシー要素: 条件](#)」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

- 多要素認証 (MFA) を要求する - AWS アカウント内の IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「[MFA 保護 API アクセスの設定](#)」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

ElastiCache コンソールの使用

Amazon ElastiCache コンソールにアクセスするには、アクセス許可の最小限のセットが必要です。これらのアクセス許可により、AWS アカウントの ElastiCache リソースの詳細をリストおよび表示できます。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き ElastiCache 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 ElastiCache 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「[ユーザーへの権限の追加](#)」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラマ的に、このアクションを完了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:ListGroupsWithUser",
        "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 ElastiCache アイデンティティとアクセスのトラブルシューティング

次の情報は、と IAM の使用時に発生する可能性がある一般的な問題の診断 ElastiCache と修正に役立ちます。

トピック

- [でアクションを実行する権限がない ElastiCache](#)
- [iam を実行する権限がありません。PassRole](#)
- [自分の AWS アカウント以外のユーザーに自分の ElastiCache リソースへのアクセスを許可したい](#)

でアクションを実行する権限がない ElastiCache

からアクションを実行する権限がないと AWS Management Console 通知された場合は、管理者に連絡してサポートを依頼する必要があります。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

以下のエラー例は、mateojackson ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の `elasticache:GetWidget` 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
elasticache:GetWidget on resource: my-example-widget
```

この場合、Mateo は、`elasticache:GetWidget` アクションを使用して *my-example-widget* リソースへのアクセスが許可されるように、管理者にポリシーの更新を依頼します。

iam を実行する権限がありません。PassRole

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して `iam:PassRole` を渡すことができるようにする必要があります ElastiCache。

一部の AWS のサービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、という IAM ユーザーがコンソールを使用して `marymajor` でアクションを実行しようする場合に発生します ElastiCache。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント以外のユーザーに自分の ElastiCache リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 ElastiCache をサポートしているかどうかを確認するには、「」を参照してください [Amazon と IAM ElastiCache の連携方法](#)。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、[IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」](#)を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの [「サードパーティー AWS アカウント が所有する へのアクセスを提供する」](#)を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの [外部認証されたユーザーへのアクセスの提供 \(ID フェデレーション\)](#) を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの [「IAM でのクロスアカウントリソースアクセス」](#)を参照してください。

アクセスコントロール

リクエストを認証するために有効な認証情報を持つことができますが、アクセス許可がない限り、ElastiCache リソースを作成またはアクセスすることはできません。例えば、ElastiCache クラスターを作成するアクセス許可が必要です。

以下のセクションでは、 のアクセス許可を管理する方法について説明します ElastiCache。最初に概要のセクションを読むことをお勧めします。

- [ElastiCache リソースに対するアクセス許可の管理の概要](#)
- [Amazon ElastiCache でのアイデンティティベースのポリシー \(IAM ポリシー\) の使用](#)

ElastiCache リソースに対するアクセス許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースの作成またはアクセスは、許可のポリシーによって管理されます。アカウント管理者は、IAM アイデンティティ (ユーザー、グループ、ロール) にアクセス許可ポリシーをアタッチできます。さらに、Amazon ElastiCache では、アクセス許可ポリシーをリソースにアタッチすることもできます。

Note

アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、「IAM ユーザーガイド」の「[IAM のベストプラクティス](#)」を参照してください。

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

- AWS IAM Identity Center のユーザーとグループ:

アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可一式を作成](#)」の手順を実行します。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

- IAM ユーザー:

- ユーザーに設定できるロールを作成します。手順については、「IAM ユーザーガイド」の「[IAM ユーザー用ロールの作成](#)」を参照してください。

- (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。「IAM ユーザーガイド」の「[ユーザー \(コンソール\) へのアクセス許可の追加](#)」の指示に従います。

トピック

- [Amazon ElastiCache のリソースとオペレーション](#)
- [リソース所有権について](#)
- [リソースへのアクセスの管理](#)
- [Amazon ElastiCache の AWS マネージドポリシー](#)
- [Amazon ElastiCache でのアイデンティティベースのポリシー \(IAM ポリシー\) の使用](#)

- [リソースレベルのアクセス許可](#)
- [条件キーの使用](#)
- [Amazon ElastiCache でのサービスにリンクされたロールの使用](#)
- [ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス](#)

Amazon ElastiCache のリソースとオペレーション

ElastiCache リソースのタイプとその ARN のリストを確認するには、「サービス認証リファレンス」の「[Amazon ElastiCache で定義されるリソース](#)」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「[Amazon ElastiCache で定義されるアクション](#)」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、リソースを作成した AWS アカウントです。つまり、リソース所有者は、そのリソースを作成するリクエストを認証するプリンシパルエンティティの AWS アカウントです。プリンシパルエンティティ はルートアカウント、IAM ユーザー、または IAM ロールです。次の例は、この仕組みを示しています。

- AWS アカウントのルートアカウント認証情報を使用してキャッシュクラスターを作成するとします。この場合、AWS アカウントはリソースの所有者です。ElastiCache では、リソースはキャッシュクラスターです。
- AWS アカウントに IAM ユーザーを作成し、キャッシュクラスターを作成するためのアクセス許可をそのユーザーに付与するとします。この場合、ユーザーはキャッシュクラスターを作成できます。ただし、キャッシュクラスターリソースを所有しているのは、このユーザーが属する AWS アカウントです。
- キャッシュクラスターを作成するためのアクセス許可のある AWS アカウントに IAM ロールを作成するとします。この場合、ロールを引き受けることができるいずれのユーザーもキャッシュクラスターを作成できます。ロールが属する AWS アカウントがキャッシュクラスターリソースを所有しています。

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 ElastiCache のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。完全な IAM ドキュメンテーションについては、「IAM ユーザーガイド」の「[IAM とは](#)」を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの [AWS IAM ポリシーの参照](#)を参照してください。

IAM アイデンティティにアタッチされているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースに添付されたポリシーは、リソースベースのポリシーと呼ばれます。

トピック

- [アイデンティティベースのポリシー \(IAM ポリシー\)](#)
- [ポリシー要素の指定: アクション、効果、リソース、プリンシパル](#)
- [ポリシーでの条件の指定](#)

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。例えば、次のオペレーションを実行できます。

- アカウントのユーザーまたはグループにアクセス許可ポリシーをアタッチする – アカウント管理者は、特定のユーザーに関連付けられるアクセス許可ポリシーを使用して、アクセス許可を付与できます。この場合、アクセス許可は、そのユーザーがキャッシュクラスター、パラメータグループ、セキュリティグループなどの ElastiCache リソースを作成するためのものです。
- 許可ポリシーをロールにアタッチする (クロスアカウントの許可を付与) - ID ベースのアクセス許可ポリシーを IAM ロールにアタッチして、クロスアカウントの許可を付与することができます。例えば、アカウント A の管理者は、次のように別の AWS アカウント (例えば、アカウント B) または AWS サービスにクロスアカウントアクセス許可を付与するロールを作成できます。
 1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースにアクセス許可を付与するロールにアクセス許可ポリシーをアタッチします。
 2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。

3. アカウント B の管理者は、ロールを引き受けるためのアクセス許可をアカウント B のユーザーに委任できます。これにより、アカウント B のユーザーは、アカウント A のリソースの作成またはアクセスが許可されます。場合によっては、AWS のサービスにロールを引き受けるためのアクセス許可を付与する必要があります。このアプローチをサポートするために、信頼ポリシーのプリンシパルを AWS のサービスのプリンシパルにすることもできます。

IAM を使用した許可の委任の詳細については、「IAM ユーザーガイド」の「[アクセス管理](#)」を参照してください。

以下に示しているのは、お客様の AWS アカウントに対する DescribeCacheClusters アクションの実行をユーザーに許可するポリシーの例です。ElastiCache では、API アクションのリソース ARN を使用した特定のリソースの識別もサポートしています。(このアプローチは、リソースレベルのアクセス許可とも呼ばれます)。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "DescribeCacheClusters",
    "Effect": "Allow",
    "Action": [
      "elasticache:DescribeCacheClusters"],
    "Resource": resource-arn
  ]
}
```

ElastiCache 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 ElastiCache でのアイデンティティベースのポリシー \(IAM ポリシー\) の使用](#)」を参照してください。ユーザー、グループ、ロール、アクセス許可の詳細については、IAM ユーザーガイドの「[アイデンティティ \(ユーザー、グループ、ロール\)](#)」を参照してください。

ポリシー要素の指定: アクション、効果、リソース、プリンシパル

サービスは、Amazon ElastiCache リソースごとに ([Amazon ElastiCache のリソースとオペレーション](#) を参照)、一連の API オペレーションを定義します ([アクション](#) を参照)。こうした API オペレーションへのアクセス権限を付与するために、ElastiCache は一連のアクションをポリシーに定義します。例えば、ElastiCache クラスターリソースに対して、アクション CreateCacheCluster、DeleteCacheCluster、DescribeCacheCluster を定義します。1 つの API オペレーションの実行で、複数のアクションのアクセス権限が必要になる場合があります。

最も基本的なポリシーの要素を次に示します。

- リソース - ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「[Amazon ElastiCache のリソースとオペレーション](#)」を参照してください。
- アクション - アクション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。たとえば、指定した Effect に応じて、elasticache:CreateCacheCluster アクセス許可では、Amazon ElastiCache CreateCacheCluster オペレーションの実行をユーザーに許可または拒否します。
- 効果 - ユーザーが特定のアクションを要求する際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に付与 (許可) していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。たとえば、別のポリシーでリソースへのアクセスが許可されているユーザーに対して、そのリソースへのアクセスを禁止できます。
- プリンシパル - ID ベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、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。

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 ElastiCache API アクションを示す表については、「[ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス](#)」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。ElastiCache 固有の条件キーを使用するには、「[条件キーの使用](#)」を参照してください。必要に応じて使用できる AWS 全体の条件キーがあります。AWS 全体を対象とするすべてのキーのリストについては、「IAM ユーザーガイド」の「[条件に利用可能なキー](#)」を参照してください。

Amazon ElastiCache の AWS マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースで権限を提供できるように設計されているため、ユーザー、グループ、ロールへの権限の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権の権限を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別に[カスタマー管理ポリシー](#)を定義することで、権限を絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されている権限を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS のサービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: ElastiCacheServiceRolePolicy

IAM エンティティに ElastiCacheServiceRolePolicy をアタッチすることはできません。このポリシーは、ユーザーに代わって ElastiCache が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。

このポリシーにより、ElastiCache はユーザーに代わってキャッシュを管理するために必要な AWS リソースを管理することができます。

- ec2 — VPC エンドポイント (サーバーレスキャッシュ用)、Elastic Network Interface (ENI) (独自設計型クラスター用)、セキュリティグループなど、キャッシュノードに接続する EC2 ネットワークリソースを管理します。
- cloudwatch — サービスから CloudWatch にメトリクスデータを送信します。
- outposts — AWS Outpost でのキャッシュノードの作成を許可します。

[ElastiCacheServiceRolePolicy](#) ポリシーは IAM コンソールにあり、[ElastiCacheServiceRolePolicy](#) は「AWS マネージドポリシーリファレンスガイド」にあります。

AWS マネージドポリシー: AmazonElastiCacheFullAccess

AmazonElastiCacheFullAccess ポリシーは IAM ID に添付できます。

このポリシーにより、プリンシパルは AWS マネジメントコンソールを使用して ElastiCache にフルアクセスできます。

- elasticache — すべての API にアクセスします。
- iam — サービスの運用に必要なサービスリンクロールを作成します。
- ec2 — キャッシュ作成に必要な依存型 EC2 リソース (VPC、サブネット、セキュリティグループ) を記述し、VPC エンドポイント (サーバーレスキャッシュ用) の作成を許可します。
- kms — カスタマー管理の CMK を保管時の暗号化に使用できるようにします。
- cloudwatch — コンソールに ElastiCache メトリクスを表示するためのメトリクスへのアクセスを許可します。
- application-autoscaling — キャッシュの自動スケーリングポリシーを記述するためのアクセスを許可します。
- logs — コンソールのログ配信機能のログストリームを入力するために使用されます。
- firehose — コンソールのログ配信機能の配信ストリームを入力するために使用されます。
- s3 — コンソールのスナップショット復元機能用の S3 バケットを入力するために使用されます。
- outposts — コンソールでキャッシュを作成するための AWS Outpost を入力するために使用されます。
- sns — コンソールの通知機能用の SNS トピックの入力に使用されます。

[AmazonElastiCacheFullAccess](#) ポリシーは IAM コンソールに、[AmazonElastiCacheFullAccess](#) は「AWS マネージドポリシーリファレンスガイド」に記載されています。

AWS マネージドポリシー: AmazonElastiCacheReadOnlyAccess

AmazonElastiCacheReadOnlyAccess ポリシーは IAM ID に添付できます。

このポリシーは、プリンシパルが AWS マネジメントコンソールを使用して ElastiCache に読み取り専用でアクセスすることを許可します。

- elasticache — 読み取り専用 Describe API にアクセスできます。

[AmazonElastiCacheReadOnlyAccess](#) ポリシーは IAM コンソールに、[AmazonElastiCacheReadOnlyAccess](#) は「AWS マネージドポリシーリファレンスガイド」に記載されています。

ElastiCache の AWS マネージドポリシーに関する更新

ElastiCache の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について確認できます。このページの変更に関する自動通知については、「ElastiCache ドキュメント履歴」ページの RSS フィードをサブスクライブしてください。

変更	説明	日付
AmazonElastiCacheFullAccess — 既存のポリシーの更新	ElastiCache に、サーバーレスキャッシュを管理し、コンソールからすべてのサービス機能を使用できるようにする新しいアクセス許可が追加されました。	2023 年 11 月 27 日
ElastiCacheServiceRolePolicy — 既存のポリシーの更新	ElastiCache に、サーバーレスキャッシュリソースの VPC エンドポイントを管理するための新しいアクセス許可が追加されました。	2023 年 11 月 27 日
ElastiCache が変更の追跡を開始	ElastiCache が AWS マネージドポリシーの変更の追跡を開始しました。	2020 年 2 月 7 日

Amazon ElastiCache でのアイデンティティベースのポリシー (IAM ポリシー) の使用

このトピックでは、アカウント管理者が IAM ID (ユーザー、グループ、ロール) へのアクセス許可ポリシーをアタッチする、ID ベースのポリシーの例を示します。

⚠ Important

最初に、Amazon ElastiCache リソースへのアクセスを管理するための基本的な概念とオプションについて説明するトピックを読むことをお勧めします。詳細については、「[ElastiCache リソースに対するアクセス許可の管理の概要](#)」を参照してください。

このセクションでは、次のトピックを対象としています。

- [Amazon ElastiCache の AWS マネージドポリシー](#)
- [カスタマーマネージドポリシーの例](#)

以下に示しているのは、アクセス許可ポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowClusterPermissions",
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateServerlessCache",
        "elasticache:CreateCacheCluster",
        "elasticache:DescribeServerlessCaches",
        "elasticache:DescribeReplicationGroups",
        "elasticache:DescribeCacheClusters",
        "elasticache:ModifyServerlessCache",
        "elasticache:ModifyReplicationGroup",
        "elasticache:ModifyCacheCluster"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUserToPassRole",
      "Effect": "Allow",
      "Action": [ "iam:PassRole" ],
      "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
    }
  ]
}
```


このポリシーには以下の 2 つのステートメントがあります。

- 最初のステートメントは、Amazon ElastiCache アクション (elasticache:Create*、elasticache:Describe*、elasticache:Modify*) のアクセス許可を付与します。
- 2 番目のステートメントは、Resource 値の最後に指定した IAM ロール名での IAM アクション iam:PassRole のアクセス許可を付与します。

ID ベースのポリシーでアクセス許可を得るプリンシパルを指定していないため、ポリシーでは Principal 要素を指定していません。ユーザーにポリシーをアタッチすると、そのユーザーが暗黙のプリンシパルになります。IAM ロールにアクセス権限ポリシーをアタッチすると、ロールの信頼ポリシーで識別されたプリンシパルがアクセス権限を得ることになります。

すべての Amazon ElastiCache API アクションとそれらが適用されるリソースの表については、「[ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス](#)」を参照してください。

カスタマーマネージドポリシーの例

デフォルトポリシーを使用せず、カスタム管理ポリシーを使用することを選択した場合は、以下の 2 点のいずれかを確認してください。iam:createServiceLinkedRole を呼び出すためのアクセス許可があることが必要です (詳細については、「[例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する](#)」を参照)。または、ElastiCache サービスにリンクされたロールを作成済みであることが必要です。

Amazon ElastiCache コンソールを使用するために必要な最小限のアクセス権限と組み合わせて、このセクションでのポリシーの例は、追加のアクセス権限を付与します。この例は、AWS SDK と AWS CLI に関連しています。

IAM ユーザーおよびグループのセットアップ手順については、IAM ユーザーガイドの「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。

Important

IAM ポリシーは必ず、本稼働環境での使用前にテストしてください。ElastiCache のアクションによっては、シンプルに見えても、ElastiCache コンソールの使用時にそれらのアクションをサポートするために、他のアクションが必要になる場合があります。たとえば、elasticache:CreateCacheCluster は、ElastiCache キャッシュクラスターを作

成するためのアクセス許可を付与します。ただし、このオペレーションを実行するために、ElastiCache コンソールでは Describe と List の多数のアクションが使用されて、リストが事前設定されます。

例

- [例 1: ユーザーに ElastiCache リソースへの読み取り専用アクセスを許可する](#)
- [例 2: ユーザーに一般的な ElastiCache システム管理者タスクの実行を許可する](#)
- [例 3: ユーザーにすべての ElastiCache API アクションへのアクセスを許可する](#)
- [例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する](#)
- [例 5: ユーザーが IAM 認証を使用してサーバーレスキャッシュに接続することを許可する](#)

例 1: ユーザーに ElastiCache リソースへの読み取り専用アクセスを許可する

以下のポリシーでは、リソースを一覧表示する ElastiCache アクションを実行するためのアクセス許可をユーザーに付与します。通常、このタイプのアクセス権限ポリシーは管理者グループにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECReadOnly",
    "Effect": "Allow",
    "Action": [
      "elasticache:Describe*",
      "elasticache:List*"
    ],
    "Resource": "*"
  ]
}
```

例 2: ユーザーに一般的な ElastiCache システム管理者タスクの実行を許可する

一般的なシステム管理者タスクには、リソースの変更が含まれます。システム管理者は ElastiCache イベントに関する情報を取得することが必要になる場合もあります。以下のポリシーでは、これらの一般的なシステム管理タスクに必要な ElastiCache アクションを実行するためのアクセス権限をユーザーに付与します。通常、このタイプのアクセス権限ポリシーはシステム管理者グループにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECAAllowMutations",
    "Effect": "Allow",
    "Action": [
      "elasticache:Modify*",
      "elasticache:Describe*",
      "elasticache:ResetCacheParameterGroup"
    ],
    "Resource": "*"
  }
]
```

例 3: ユーザーにすべての ElastiCache API アクションへのアクセスを許可する

以下のポリシーでは、ユーザーにすべての ElastiCache アクションへのアクセスを許可します。このタイプのアクセス権限ポリシーは管理者ユーザーにのみ付与することをお勧めします。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECAAllowAll",
    "Effect": "Allow",
    "Action": [
      "elasticache:*"
    ],
    "Resource": "*"
  }
]
```

例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する

次のポリシーでは、ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可します。mutative ElastiCache オペレーションを実行するユーザーには、このタイプのアクセス許可ポリシーを与えることをお勧めします。

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

```
{
  "Sid": "CreateSLRAllows",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "iam:AWSServiceName": "elasticache.amazonaws.com"
    }
  }
}
```

例 5: ユーザーが IAM 認証を使用してサーバーレスキャッシュに接続することを許可する

次のポリシーでは、どのユーザーにも、IAM 認証を使用して 2023 年 4 月 1 日から 2023 年 6 月 30 日までのサーバーレスキャッシュに接続することを許可します。

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:*"
      ],
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2023-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2023-06-30T23:59:59Z"}
      }
    },
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:user:*"
      ]
    }
  ]
}
```

```
]
}
```

リソースレベルのアクセス許可

IAM ポリシーでリソースを指定することで、アクセス許可の範囲を制限できます。ElastiCache API アクションの多くは、アクションの動作に応じて異なるリソースタイプをサポートしています。各 IAM ポリシーステートメントによって、リソースで実行されるアクションに対するアクセス許可が付与されます。アクションが名前の付いたリソースで動作しない場合、またはすべてのリソースに対してアクションを実行するアクセス許可を付与した場合、ポリシー内のリソースの値はワイルドカード (*) になります。多くの API アクションでは、リソースの Amazon リソースネーム (ARN)、または複数のリソースに一致する ARN パターンを指定することによって、ユーザーが変更できるリソースを制限できます。リソース別にアクセス許可を制限するには、ARN 別にリソースを指定します。

ElastiCache リソースのタイプとその ARN のリストを確認するには、「サービス認証リファレンス」の「[Amazon ElastiCache で定義されるリソース](#)」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「[Amazon ElastiCache で定義されるアクション](#)」を参照してください。

例

- [例 1: 特定の ElastiCache リソースタイプへのフルアクセスをユーザーに許可する](#)
- [例 2: サーバーレスキャッシュへのユーザーアクセスを拒否する。](#)

例 1: 特定の ElastiCache リソースタイプへのフルアクセスをユーザーに許可する

次のポリシーでは、すべてのリソースタイプのサーバーレスキャッシュを明示的に許可します。

```
{
  "Sid": "Example1",
  "Effect": "Allow",
  "Action": "elasticache:*",
  "Resource": [
    "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
  ]
}
```

例 2: サーバーレスキャッシュへのユーザーアクセスを拒否する。

次の例では、特定のサーバーレスキャッシュへのアクセスを明示的に拒否します。

```
{
  "Sid": "Example2",
  "Effect": "Deny",
  "Action": "elasticache:*",
  "Resource": [
    "arn:aws:elasticache:us-east-1:account-id:serverlesscache:name"
  ]
}
```

条件キーの使用

IAM ポリシーを有効にする方法を決める条件を指定できます。ElastiCache では、JSON ポリシーの Condition 要素を使用して、リクエストコンテキストのキーを、ポリシーで指定したキー値と比較できます。ポリシー要素の詳細については、[IAM JSON policy elements: Condition](#) を参照してください。

ElastiCache の条件キーのリストを確認するには、「サービス認証リファレンス」の「[Amazon ElastiCache の条件キー](#)」を参照してください。

グローバル条件キーのリストについては、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

条件の指定: 条件キーの使用

きめ細かなコントロールを実装するには、特定のリクエストに対して個別のパラメータセットを制御するための条件を指定した IAM アクセス許可ポリシーを作成します。次に、IAM コンソールを使用して作成する IAM ユーザー、グループ、またはロールにそのポリシーを適用します。

条件を適用するには、条件情報を IAM ポリシーステートメントに追加します。次の例では、独自に設計されたすべてのキャッシュクラスターがノードタイプ `cache.r5.large` になるという条件を指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
    }
  ],
}
```

```
    "Resource": [
      "arn:aws:elasticache:*:*:parametergroup:*",
      "arn:aws:elasticache:*:*:subnetgroup:*"
    ],
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:cluster:*",
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "StringEquals": {
        "elasticache:CacheNodeType": [
          "cache.r5.large"
        ]
      }
    }
  }
]
}
```

詳細については、「[タグベースのアクセスコントロールポリシーの例](#)」を参照してください。

ポリシー条件演算子の使用に関する詳細については、「[ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス](#)」を参照してください。

ポリシー例: きめ細かなパラメータコントロールのための IAM ポリシー条件の使用

このセクションでは、前述の ElastiCache パラメータに対してきめ細かなアクセスコントロールを実装するためのポリシー例について説明します。

1. `elasticache:MaximumDataStorage`: サーバーレスキャッシュの最大データストレージを指定します。指定された条件を使用して、特定の量を超えるデータを保存できるキャッシュを作成することはできません。

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

```

    {
      "Sid": "AllowDependentResources",
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateServerlessCache"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
        "arn:aws:elasticache:*:*:snapshot:*",
        "arn:aws:elasticache:*:*:usergroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateServerlessCache"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:serverlesscache:*"
      ],
      "Condition": {
        "NumericLessThanEquals": {
          "elasticache:MaximumDataStorage": "30"
        },
        "StringEquals": {
          "elasticache:DataStorageUnit": "GB"
        }
      }
    }
  ]
}

```

2. `elasticache:MaximumECPUPerSecond`: サーバーレスキャッシュの 1 秒あたりの最大 ECPU 値を指定します。指定された条件では、1 秒あたりに特定の数を超える ECPU を実行できるキャッシュを作成することはできません。

```

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

```



```

        "elasticache:CreateServerlessCache"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
        "arn:aws:elasticache:*:*:snapshot:*",
        "arn:aws:elasticache:*:*:usergroup:*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "elasticache:CreateServerlessCache"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:serverlesscache:*"
    ],
    "Condition": {
        "NumericLessThanEquals": {
            "elasticache:MaximumECPUPerSecond": "100000"
        }
    }
}
]
}

```

3. `elasticache:CacheNodeType`: ユーザーが作成できる `NodeType` を指定します。指定された条件を使用して、ノードタイプの単一値または範囲値を指定できます。

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
    ],
}

```

```
{
  "Effect": "Allow",
  "Action": [
    "elasticache:CreateCacheCluster",
    "elasticache:CreateReplicationGroup"
  ],
  "Resource": [
    "arn:aws:elasticache:*:*:cluster:*",
    "arn:aws:elasticache:*:*:replicationgroup:*"
  ],
  "Condition": {
    "StringEquals": {
      "elasticache:CacheNodeType": [
        "cache.t2.micro",
        "cache.t2.medium"
      ]
    }
  }
}
```

4. elasticache:NumNodeGroups: 20 未満のノードグループを持つレプリケーショングループを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ]
    }
  ]
}
```

```
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "NumericLessThanEquals": {
        "elasticache:NumNodeGroups": "20"
      }
    }
  }
]
```

5. `elasticache:ReplicasPerNodeGroup`: ノードごとのレプリカを 5~10 の間で指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "NumericGreaterThanEquals": {
          "elasticache:ReplicasPerNodeGroup": "5"
        },
        "NumericLessThanEquals": {
          "elasticache:ReplicasPerNodeGroup": "10"
        }
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

6. `elasticache:EngineVersion`: バージョン 5.0.6 の使用量を指定します。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "elasticache:CreateCacheCluster",  
        "elasticache:CreateReplicationGroup"  
      ],  
      "Resource": [  
        "arn:aws:elasticache:*:*:parametergroup:*",  
        "arn:aws:elasticache:*:*:subnetgroup:*"  
      ]  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "elasticache:CreateCacheCluster",  
        "elasticache:CreateReplicationGroup"  
      ],  
      "Resource": [  
        "arn:aws:elasticache:*:*:cluster:*",  
        "arn:aws:elasticache:*:*:replicationgroup:*"  
      ],  
      "Condition": {  
        "StringEquals": {  
          "elasticache:EngineVersion": "5.0.6"  
        }  
      }  
    }  
  ]  
}
```

7. `elasticache:EngineType`: Redis エンジンのみを使用を指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*",
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticache:EngineType": "redis"
        }
      }
    }
  ]
}
```

8. `elasticache:AtRestEncryptionEnabled`: レプリケーショングループが暗号化を有効にしてのみ作成されるように指定します。

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

```

    "Effect": "Allow",
    "Action": [
        "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
        "Bool": {
            "elasticache:AtRestEncryptionEnabled": "true"
        }
    }
}
]
}

```

9. elasticache:TransitEncryptionEnabled

- a. [CreateReplicationGroup](#) アクションの `elasticache:TransitEncryptionEnabled` 条件キーを `false` に設定して、TLS を使用していない場合にのみレプリケーショングループを作成できるように指定します。

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        }
    ]
}

```

```

    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "Bool": {
        "elasticache:TransitEncryptionEnabled": "false"
      }
    }
  }
]
}

```

[CreateReplicationGroup](#) アクションのポリシーで

`elasticache:TransitEncryptionEnabled` 条件キーが `false` に設定されている場合、TLS が使用されていない (つまり、リクエストで `TransitEncryptionEnabled` パラメータが `true` に設定されていない、または `TransitEncryptionMode` パラメータが `required` に設定されている) 場合のみ `CreateReplicationGroup` リクエストが許可されます。

- b. [CreateReplicationGroup](#) アクションの `elasticache:TransitEncryptionEnabled` 条件キーを `true` に設定して、TLS が使用されている場合にのみレプリケーショングループを作成できるように指定します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    }
  ]
}

```

```

    ],
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "Bool": {
        "elasticache:TransitEncryptionEnabled": "true"
      }
    }
  }
]
}

```

[CreateReplicationGroup](#) アクションのポリシーで

`elasticache:TransitEncryptionEnabled` 条件キーが `true` に設定されている場合、リクエストで `TransitEncryptionEnabled` パラメータが `true` に設定されていて、`TransitEncryptionMode` パラメータが `required` に設定されている場合のみ `CreateReplicationGroup` リクエストが許可されます。

- c. `ModifyReplicationGroup` アクションで、TLS が使用されている場合にのみレプリケーショングループを変更できるように指定するには、`elasticache:TransitEncryptionEnabled` を `true` に設定します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:ModifyReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {

```



```

        "BoolIfExists": {
            "elasticache:TransitEncryptionEnabled": "true"
        }
    }
}
]
}

```

[CreateReplicationGroup](#) アクションのポリシーで

`elasticache:TransitEncryptionEnabled` 条件キーが `true` に設定されている場合、リクエストで `TransitEncryptionMode` パラメータが `required` に設定されている場合のみ `ModifyReplicationGroup` リクエストが許可されます。`true` に設定した `TransitEncryptionEnabled` パラメータもオプションで含めることができますが、これは TLS を有効にするには必要ありません。

10 `elasticache:AutomaticFailoverEnabled`: レプリケーショングループが自動フェイルオーバーを有効にしてのみ作成されるように指定します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "Bool": {

```

```
        "elasticache:AutomaticFailoverEnabled": "true"
    }
}
}
```

11 `elasticache:MultiAZEnabled`: レプリケーショングループがマルチ AZ を無効にしては作成できないように指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*",
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "Bool": {
          "elasticache:MultiAZEnabled": "false"
        }
      }
    }
  ]
}
```

12.elasticache:ClusterModeEnabled: レプリケーショングループがクラスターモードを有効にしてのみ作成できるように指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "Bool": {
          "elasticache:ClusterModeEnabled": "true"
        }
      }
    }
  ]
}
```

13.elasticache:AuthTokenEnabled: レプリケーショングループが AUTH トークンを有効にしてのみ作成できるように指定します。

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

```

    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:parametergroup:*",
      "arn:aws:elasticache:*:*:subnetgroup:*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:CreateReplicationGroup"
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:cluster:*",
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "Bool": {
        "elasticache:AuthTokenEnabled": "true"
      }
    }
  }
]
}

```

14 `elasticache:SnapshotRetentionLimit`: スナップショットを保持する日数 (または最小/最大) を指定します。以下のポリシーは、少なくとも 30 日間バックアップを保存することを強制します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [

```

```

        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup",
        "elasticache:CreateServerlessCache"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:cluster:*",
        "arn:aws:elasticache:*:*:replicationgroup:*",
        "arn:aws:elasticache:*:*:serverlesscache:*"
    ],
    "Condition": {
        "NumericGreaterThanEquals": {
            "elasticache:SnapshotRetentionLimit": "30"
        }
    }
}
]
}

```

15 elasticache:KmsKeyId: 顧客管理の AWS KMS キーの使用量を指定します

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {

```

```

    "Effect": "Allow",
    "Action": [
        "elasticache:CreateServerlessCache"
    ],
    "Resource": [
        "arn:aws:elasticache:*:*:serverlesscache:*"
    ],
    "Condition": {
        "StringEquals": {
            "elasticache:KmsKeyId": "my-key"
        }
    }
}
]
}

```

16 `elasticache:CacheParameterGroupName`: クラスター上の組織の特定のパラメータを使用して、デフォルト以外のパラメータグループを指定します。パラメータグループの命名パターンを指定したり、特定のパラメータグループ名に対するブロック削除を指定することもできます。以下は、「my-org-param-group」のみの使用量を制限する例です。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],

```

```

    "Resource": [
      "arn:aws:elasticache:*:*:cluster:*",
      "arn:aws:elasticache:*:*:replicationgroup:*"
    ],
    "Condition": {
      "StringEquals": {
        "elasticache:CacheParameterGroupName": "my-org-param-group"
      }
    }
  }
]
}

```

17 elasticache:CreateCacheCluster: リクエストタグ Project が欠落しているか、Dev、QA、または Prod と等しくない場合、CreateCacheCluster アクションが拒否されます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*",
        "arn:aws:elasticache:*:*:securitygroup:*",
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/Project": "true"
        }
      }
    }
  ]
}

```

```

    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:AddTagsToResource"
    ],
    "Resource": "arn:aws:elasticache:*:*:cluster:*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/Project": [
          "Dev",
          "Prod",
          "QA"
        ]
      }
    }
  }
]
}

```

18 `elasticache:CacheNodeType: cacheNodeType cache.r5.large` または `cache.r6g.4xlarge` を使用した `CreateCacheCluster` およびタグ `Project=XYZ` を許可します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ]
    }
  ]
}

```



```
    ],
    "Resource": [
      "arn:aws:elasticache:*:*:cluster:*"
    ],
    "Condition": {
      "StringEqualsIfExists": {
        "elasticache:CacheNodeType": [
          "cache.r5.large",
          "cache.r6g.4xlarge"
        ]
      },
      "StringEquals": {
        "aws:RequestTag/Project": "XYZ"
      }
    }
  }
}
```

Note

タグやその他の条件キーと一緒に強制するポリシーを作成する際は、条件付き IfExists は、--tags パラメータを用いた作成リクエストの追加の elasticache:AddTagsToResource ポリシー要件が原因で、条件キー要素で必要となる場合があります。

Amazon ElastiCache でのサービスにリンクされたロールの使用

Amazon ElastiCache は、AWS Identity and Access Management (IAM) の [サービスにリンクされたロール](#) を使用しています。サービスにリンクされたロールは、Amazon ElastiCache などの AWS サービスに直接リンクされた一意のタイプの IAM ロールです。Amazon ElastiCache サービスにリンクされたロールは、Amazon ElastiCache によって事前定義されています。それらには、サービスがユーザーのクラスターに代わって AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれます。

サービスにリンクされたロールを使用することで、必要なアクセス権限を手動で追加する必要がなくなるため、Amazon ElastiCache の設定が簡単になります。ロールは AWS アカウント内にありますが、Amazon ElastiCache のユースケースにリンクされており、アクセス権限が事前に定義されています。これらのロールを引き受けることができるのは Amazon ElastiCache のみで、事前定義され

たアクセス許可ポリシーを使用することができるのはこれらのロールのみです。ロールを削除するには、まず関連リソースを削除します。これにより、リソースにアクセスするのに必要なアクセス許可を誤って削除することがなくなり、Amazon ElastiCache リソースは保護されます。

サービスリンクロールをサポートする他のサービスについては、[IAM と連携する AWS のサービス](#)を参照して、[Service-linked role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを見つけてください。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、[Yes] (はい) リンクを選択します。

目次

- [Amazon ElastiCache でのサービスにリンクされたロールのアクセス許可](#)
 - [サービスリンクロールを作成するためのアクセス許可](#)
- [サービスにリンクされたロールの作成 \(IAM\)](#)
 - [サービスにリンクされたロールの作成 \(IAM コンソール\)](#)
 - [サービスにリンクされたロールの作成 \(IAM CLI\)](#)
 - [サービスにリンクされたロールの作成 \(IAM API\)](#)
- [Amazon ElastiCache のサービスにリンクされたロールの説明の編集](#)
 - [サービスにリンクされたロールの説明の編集 \(IAMコンソール\)](#)
 - [サービスにリンクされたロールの説明の編集 \(IAM CLI\)](#)
 - [サービスにリンクされたロールの説明の編集 \(IAM API\)](#)
- [Amazon ElastiCache でのサービスにリンクされたロールの削除](#)
 - [サービスにリンクされたロールのクリーンアップ](#)
 - [サービスにリンクされたロールの削除 \(IAMコンソール\)](#)
 - [サービスにリンクされたロールの削除 \(IAM CLI\)](#)
 - [サービスにリンクされたロールの削除 \(IAM API\)](#)

Amazon ElastiCache でのサービスにリンクされたロールのアクセス許可

サービスリンクロールを作成するためのアクセス許可

IAM エンティティが AWS ServiceRoleForElastiCache サービスリンクロールを作成することを許可するには

以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
```

```
"Effect": "Allow",
"Action": [
  "iam:CreateServiceLinkedRole",
  "iam:PutRolePolicy"
],
"Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWSServiceRoleForElastiCache*",
"Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

IAM エンティティが AWS ServiceRoleForElastiCache サービスリンクロールを削除することを許可するには

以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWSServiceRoleForElastiCache*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

または AWS 管理ポリシーを使用して、Amazon ElastiCache へのフルアクセスを許可することもできます。

サービスにリンクされたロールの作成 (IAM)

IAM コンソール、CLI または API を使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールの作成 (IAM コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 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、新しいロールの作成を選択します。

3. [Select type of trusted entity] (信頼されたエンティティの種類を選択) の下で、[AWS Service] (サービス) を選択します。
4. [Or select a service to view its use cases] (またはサービスを選択してそのユースケースを表示) で、[ElastiCache] を選択します。
5. [Next: Permissions] (次のステップ: アクセス許可) を選択します。
6. ポリシー名 の下で、ElastiCacheServiceRolePolicy はこのロールに必要なことに注意してください。次: タグ を選択します。
7. タグは、サービスにリンクされたロールではサポートされないことに注意してください。次: レビュー を選択します。
8. 「オプション」ロールの説明 で、サービスにリンクされた新しいロールの説明を編集します。
9. ロール情報を確認し、ロールの作成 を選択します。

サービスにリンクされたロールの作成 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスに必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (CLI)

次のオペレーションを使用してください。

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

サービスにリンクされたロールの作成 (IAM API)

IAM API を使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスに必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (API)

[CreateServiceLinkedRole](#) API コールを使用します。リクエストで、サービス名 `elasticache.amazonaws.com` を指定します。

Amazon ElastiCache のサービスにリンクされたロールの説明の編集

Amazon ElastiCache では、AWSServiceRoleForElastiCache のサービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロール

が参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。

サービスにリンクされたロールの説明の編集 (IAMコンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

サービスにリンクされたロールの説明を編集するには (コンソール)

1. IAM コンソールのナビゲーションペインで [ロール] を選択します。
2. 変更するロールの名前を選択します。
3. ロールの説明の右端にある編集を選択します。
4. ボックスに新しい説明を入力し、保存を選択します。

サービスにリンクされたロールの説明の編集 (IAM CLI)

サービスにリンクされたロールの説明は、AWS Command Line Interface から IAM オペレーションを使用して編集できます。

サービスにリンクされたロールの説明を変更するには (CLI)

1. 「オプション」現在のロールの説明を表示するには、IAM オペレーション [get-role](#) の AWS CLI を使用します。

Example

```
$ aws iam get-role --role-name AWSServiceRoleForElastiCache
```

CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。たとえば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。

2. サービスにリンクされたロールの説明を更新するには、IAM オペレーション [update-role-description](#) の AWS CLI を使用します。

Linux、macOS、Unix の場合:

```
$ aws iam update-role-description \  
  --role-name AWSServiceRoleForElastiCache \  
  --description "new description"
```

Windows の場合:

```
$ aws iam update-role-description ^  
  --role-name AWSServiceRoleForElastiCache ^  
  --description "new description"
```

サービスにリンクされたロールの説明の編集 (IAM API)

サービスにリンクされたロールの説明は、IAM API を使用して編集できます。

サービスにリンクされたロールの説明を変更するには (API)

1. 「オプション」現在のロールの説明を表示するには、IAM API オペレーション [GetRole](#) を使用します。

Example

```
https://iam.amazonaws.com/  
  ?Action=GetRole  
  &RoleName=AWSServiceRoleForElastiCache  
  &Version=2010-05-08  
  &AUTHPARAMS
```

2. ロールの説明を更新するには、IAM API オペレーション [UpdateRoleDescription](#) を使用します。

Example

```
https://iam.amazonaws.com/  
  ?Action=UpdateRoleDescription  
  &RoleName=AWSServiceRoleForElastiCache  
  &Version=2010-05-08  
  &Description="New description"
```

Amazon ElastiCache 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が必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。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ーリングされたり、メンテナンスされたりすることがなくなります。ただし、削除する前に、サービスにリンクされた役割をクリーンアップする必要があります。

Amazon ElastiCache はサービスにリンクされたロールを削除しません。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそれに関連付けられているリソース (クラスターまたはレプリケーショングループ) がないことを確認する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 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。AWSServiceRoleForElastiCache ロールのチェックボックスではなく、ロールの名前を選択します。
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。

AWSServiceRoleForElastiCache が必要な Amazon ElastiCache リソースを削除するには

- クラスターを削除するには、以下を参照してください。
 - [AWS Management Console を使用する場合](#)
 - [AWS CLIの使用](#)
 - [ElastiCache API の使用](#)
- レプリケーショングループを削除するには、以下を参照してください。
 - [レプリケーショングループの削除 \(コンソール\)](#)
 - [レプリケーショングループの削除 \(AWS CLI\)](#)
 - [レプリケーショングループの削除 \(ElastiCache API \)](#)

サービスにリンクされたロールの削除 (IAMコンソール)

IAM コンソールを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 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。ロール名または行そのものではなく、削除するロール名の横にあるチェックボックスをオンにします。
3. ページ上部にある [ロールのアクション] で [ロールの削除] を選択します。
4. 確認ダイアログボックスで、サービスの最終アクセス時間データを確認します。これは、選択したそれぞれのロールの AWS サービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。
5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送信すると、削除タスクは成功または失敗する可能性があります。タスクが失敗した場合は、通知から 詳細を表示または リソースを表示を選択して、削除が失敗した理由を知ることができます。

サービスにリンクされたロールの削除 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (CLI)

1. 削除するサービスにリンクされたロールの名前が分からない場合、以下のコマンドを入力します。このコマンドでは、アカウントにあるロールとその Amazon リソースネーム (ARN) を一覧表示します。

```
$ aws iam get-role --role-name role-name
```

CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールに ARN `arn:aws:iam::123456789012:role/myrole` がある場合、そのロールを **myrole** と参照します。

2. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `deletion-task-id` を

取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、以下を入力します。

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. 削除タスクのステータスを確認するには、以下を入力します。

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

削除タスクのステータスは、NOT_STARTED、IN_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

サービスにリンクされたロールの削除 (IAM API)

IAM API を使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (API)

1. サービスにリンクされたロールの削除リクエストを送信するには、[DeleteServiceLinkedRole](#) を呼び出します。リクエストで、ロール名を指定します。

サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから DeletionTaskId を取得して、削除タスクのステータスを確認する必要があります。

2. 削除タスクのステータスを確認するには、[GetServiceLinkedRoleDeletionStatus](#) を呼び出します。リクエストで DeletionTaskId を指定します。

削除タスクのステータスは、NOT_STARTED、IN_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス

[アクセスコントロール](#) を設定し、IAM ポリシーにアタッチするアクセス許可ポリシー (アイデンティティベースまたはリソースベース) を作成するときは、以下の表をリファレンスとして使用できます。この表には、各 Amazon ElastiCache API オペレーションと、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が一覧表示されています。ポリシーの Action フィールドでアクションを指定し、ポリシーの Resource フィールドでリソースの値を指定します。特に明記されていない限り、リソースは必須です。一部のフィールドには、必須リソースとオプションリソースの両方が含まれます。リソース ARN がない場合、ポリシー内のリソースはワイルドカード (*) になります。

ElastiCache ポリシーで条件キーを使用して条件を表現できます。ElastiCache 固有の条件キーのリストと、それらが適用されるアクションとリソースタイプを確認するには、「」を参照してください [条件キーの使用](#)。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 [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

アクションを指定するには、API オペレーション名 (elasticache:DescribeCacheClusters など) の前に elasticache: プレフィックスを使用します。

ElastiCache アクションのリストを確認するには、「サービス認証リファレンス」の「[Amazon で定義されるアクション ElastiCache](#)」を参照してください。

Amazon のコンプライアンス検証 ElastiCache


サードパーティーの監査者は、SOC、PCI、FedRAMP、HIPAA などの複数の AWS コンプライアンスプログラムの一環として、AWS サービスのセキュリティとコンプライアンスを評価します。

AWS のサービスが特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラム [AWS のサービスによる対象範囲内のコンプライアンスプログラム](#) を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS 「コンプライアンスプログラム」](#) を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「[でのレポートのダウンロード AWS Artifact](#)」の「」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービスは、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- [アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ](#) – このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

 Note

すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「[HIPAA 対応サービスのリファレンス](#)」を参照してください。

- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- [AWS カスタマーコンプライアンスガイド](#) – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめられています。
- 「[デベロッパーガイド](#)」の「[ルールによるリソースの評価](#)」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。AWS Config
- [AWS Security Hub](#) – これにより AWS のサービス、内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「[Security Hub のコントロールリファレンス](#)」を参照してください。
- [Amazon GuardDuty](#) – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。

- [AWS Audit Manager](#) – これにより AWS のサービス、AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

詳細情報

AWS クラウドコンプライアンスの一般的な情報については、以下を参照してください。

- [サービス別の FIPS エンドポイント](#)
- [でのサービスの更新 ElastiCache](#)
- [AWS クラウドコンプライアンス](#)
- [責任共有モデル](#)
- [AWS PCI DSS コンプライアンスプログラム](#)

Amazon ElastiCache s の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心に構築されます。AWS リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「[AWS グローバルインフラストラクチャ](#)」を参照してください。

Amazon ElastiCache では、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に対応できるように複数の機能を提供しています。

トピック

- [障害の軽減](#)

障害の軽減

Amazon ElastiCache の実装を計画するときは、障害がアプリケーションやデータに与える影響を最小限に抑えるように計画する必要があります。このセクションのトピックでは、アプリケーションおよびデータを障害から保護するために実行できるアプローチについて説明します。

トピック

- [Redis 実行時の障害の軽減](#)
- [レコメンデーション](#)

Redis 実行時の障害の軽減

Redis エンジンを実行する場合に、ノードまたはアベイラビリティーゾーンの障害の影響を最小限にする方法として、次のオプションがあります。

ノードの障害の軽減

サーバーレスキャッシュは、マルチ AZ アーキテクチャでノード障害を自動的に軽減するため、ノード障害はアプリケーションにとって透過的です。個々のノードの障害を軽減するには、独自設計型クラスターを適切に設定する必要があります。

独自設計型クラスターにおける Redis ノードの障害の影響を軽減するには、次のオプションがあります。

トピック

- [障害の軽減: Redis レプリケーショングループ](#)

障害の軽減: Redis レプリケーショングループ

Redis レプリケーショングループは、アプリケーションの読み取りと書き込みが可能な単一のプライマリノードと、1~5 個の読み取り専用のレプリカノードで構成されます。データがプライマリノードに書き込まれるときは、常にリードレプリカノードでデータが非同期的に更新されます。

リードレプリカが失敗した場合

1. ElastiCache は、失敗したリードレプリカを検出します。
2. ElastiCache は、障害が発生したノードをオフラインにします。
3. ElastiCache は、同じ AZ で代替ノードを起動してプロビジョニングします。

4. 新しいノードがプライマリノードと同期されます。

この間、アプリケーションは他のノードを使用して読み書きを続行できます。

Redis のマルチ AZ

Redis レプリケーショングループでマルチ AZ を有効にすることができます。マルチ AZ を有効にするかどうかにかかわらず、失敗したプライマリが検出され、自動的に置き換えられます。これを実行する方法は、マルチ AZ が有効かどうかによって異なります。

マルチ AZ が有効な場合

1. ElastiCache はプライマリノードの障害を検出します。
2. ElastiCache は、レプリケーションラグが最も少ないリードレプリカノードをプライマリノードに昇格させます。
3. 他のレプリカと新しいプライマリノードを同期します。
4. ElastiCache は、障害が発生したプライマリの AZ でリードレプリカをスピニングアップします。
5. 新しいノードが、新たに昇格されたプライマリと同期されます。

レプリカノードへのフェイルオーバーは、通常、新しいプライマリノードを作成してプロビジョニングするより高速です。つまり、マルチ AZ が有効でない場合と比べて、アプリケーションはプライマリノードへの書き込みをすばやく再開できます。

詳細については、「[マルチ AZ による ElastiCache for Redis のダウンタイムの最小化](#)」を参照してください。

マルチ AZ が無効な場合

1. ElastiCache はプライマリ障害を検出します。
2. ElastiCache はプライマリをオフラインにします。
3. ElastiCache は、障害が発生したプライマリを置き換える新しいプライマリノードを作成してプロビジョニングします。
4. ElastiCache は、新しいプライマリを既存のレプリカの 1 つと同期します。
5. 同期が終了すると、新しいノードはクラスターのプライマリノードとなります。

このプロセスのステップ 1~4 が終了するまで、アプリケーションはプライマリノードに書き込みません。ただし、アプリケーションはレプリカノードから読み込みを続けることができます。

保護を強化するために、別のアベイラビリティーゾーン (AZ) でレプリケーショングループのノードを起動することをお勧めします。これを行った場合、AZ の障害の影響をその AZ のノードのみにとどめ、他のノードには影響を与えません。

詳細については、「[レプリケーショングループを使用する高可用性](#)」を参照してください。

アベイラビリティーゾーンの障害の軽減

サーバーレスキャッシュは、レプリケートされたマルチ AZ アーキテクチャでアベイラビリティーゾーンの障害を自動的に軽減するため、AZ 障害はアプリケーションにとって透過的です。

独自設計型クラスターにおけるアベイラビリティーゾーンの障害の影響を軽減するためには、各シャードのノードを可能な限り多くのアベイラビリティーゾーンにノードを配置します。

シャードにノードがいくつあったとしても、そのすべてが同じアベイラビリティーゾーンにある場合は、その AZ で壊滅的な障害が発生するとシャードのデータのすべてが失われます。ただし、複数の AZ にノードがある場合は、いずれかの AZ で障害が発生しても失われるのはその AZ のノードのみとなります。

ノードを失った場合は、読み込みオペレーションがより少ない数のノードによって共有されるようになるため、パフォーマンスが低下します。このパフォーマンスの低下は、ノードが置き換えられるまで続きます。

Redis ノードのアベイラビリティーゾーンを指定する方法の詳細については、「[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#)」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詳細については、「[リージョンとアベイラビリティーゾーンの選択](#)」を参照してください。

レコメンデーション

追加の設定をしなくても自動的に耐障害性が向上するため、独自設計型クラスターよりもサーバーレスキャッシュを作成することをお勧めします。ただし、独自設計型クラスターを作成する場合は、個別ノードの障害と、幅広いアベイラビリティーゾーンの障害の 2 種類の障害を想定する必要があります。ベストの障害軽減プランは、両方のタイプの障害に対処します。

ノード障害の影響を最小限に抑える

ノードの障害の影響を最小限に抑えるために、各シャードに複数のノードを実装して、複数のアベイラビリティーゾーンにノードを分散することをお勧めします。これはサーバーレスキャッシュでは自動的に行われます。

独自設計型クラスターでは、レプリケーショングループでマルチ AZ を有効にすることをお勧めします。これにより、プライマリノードに障害が発生した場合、ElastiCache は自動的にレプリカにフェイルオーバーされます。

アベイラビリティゾーンの障害の影響を最小限に抑える

アベイラビリティゾーンの障害の影響を最小限に抑えるには、できるだけ多くの異なるアベイラビリティゾーンでノードを起動することをお勧めします。ノードを AZ 間に均等に分散することで、予期しない AZ の障害が発生した場合の影響を最小化します。これはサーバーレスキャッシュでは自動的に行われます。

その他の対策

Redis を実行する場合は、上記に加えてクラスターのバックアップを定期的にとることをお勧めします。バックアップ (スナップショット) によって、障害や破損が発生した場合にキャッシュを復元するのに使用できる .rdb ファイルが作成されます。詳細については、「[スナップショットおよび復元](#)」を参照してください。

AWS ElastiCache で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として、AWS ElastiCache は、[AWS アーキテクチャセンター](#)の「セキュリティとコンプライアンスセクション」で説明されている AWS グローバルネットワークセキュリティ手順によって保護されます。

AWS が公開している API コールを使用して、ネットワーク経由で ElastiCache にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。TLS 1.3 以降が推奨されます。また、一時的ディフィー・ヘルマン Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストには、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。

でのサービスの更新 ElastiCache

ElastiCache は、キャッシュ、クラスター、ノードのフリートを自動的にモニタリングして、サービスの更新が利用可能になったときにそれらを適用します。サーバーレスキャッシュのサービス更新は

自動的かつ透過的に適用されます。自己設計クラスターの場合、これらの更新 ElastiCache を適用できるように、事前定義されたメンテナンスウィンドウを設定します。ただし、場合によっては、このアプローチが厳格すぎて、ビジネスフローが制限される可能性もあります。

サービス更新では、独自設計型クラスターにアップデートを適用するタイミングと内容を制御できません。選択した ElastiCache クラスターに対するこれらの更新の進行状況をリアルタイムでモニタリングすることもできます。

サービスアップデートの管理

ElastiCache 自社設計クラスターのサービスアップデートは定期的にリリースされます。これらのサービス更新の対象となる自社設計のクラスターが 1 つ以上ある場合は、更新がリリースされると、電子メール、SNS、Personal Health Dashboard (PHD)、および Amazon CloudWatch イベントを通じて通知を受け取ります。更新はコンソールの [サービスアップデート] ページにも表示されます。ElastiCache このダッシュボードを使用すると、ElastiCache フリートのすべてのサービスアップデートとそのステータスを表示できます。サーバーレスキャッシュのサービスアップデートは透過的に適用され、[サービスの更新] では管理できません。

自動更新を開始する前に、更新を適用するタイミングを制御します。ElastiCache up-to-date クラスターに常に最新のセキュリティパッチが適用されるように、security-update タイプの更新はできるだけ早く適用することを強くお勧めします。

以下のセクションでは、これらのオプションについて詳しく説明します。

トピック

- [サービスの更新の適用](#)
- [コンソールを使用して最新のサービスアップデートが適用されていることを確認する AWS](#)
- [サービスの更新の停止](#)

サービスの更新の適用

フリートに対するサービスの更新の適用は、更新が 使用可能 ステータスになってから開始することができます。サービスの更新は累積的です。つまり、未適用の更新も最新の更新に含まれます。

サービスアップデートで自動アップデートが有効になっている場合は、利用可能になっても何も実行しないように選択できます。ElastiCache 自動更新の開始日以降に予定されているクラスターのメンテナンス期間中に更新を適用するようにスケジュールします。更新のステージごとに、関連する通知を受け取ります。

Note

ステータスが 使用可能または スケジュール済み であるサービスの更新だけを適用できません。

サービス固有の更新を確認して該当するクラスターに適用する方法の詳細については、を参照してください。ElastiCache [コンソールを使用したサービスの更新の適用](#)

1 ElastiCache つ以上のクラスターで新しいサービスアップデートが利用可能になったら、ElastiCache コンソール、API、AWS CLI またはを使用してアップデートを適用できます。次のセクションでは、更新の適用に使用できるオプションについて説明します。

コンソールを使用したサービスの更新の適用

使用可能なサービスの更新のリストと他の情報を確認するには、コンソールの サービスの更新 (サービスの更新) ページに移動します。

1. AWS Management Console にサインインし、<https://console.aws.amazon.com/elasticache/> にある Amazon ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[Service Updates] (サービスの更新) を選択します。
3. [Service updates] (サービスの更新) では、次の項目を表示できます。
 - [Service update name] (サービスの更新名): サービスの更新の一意の名前
 - [Update type] (更新タイプ): サービスの更新のタイプ ([security-update] (セキュリティ更新) または [engine-update] (エンジン更新) のいずれか)
 - [Update severity] (重大度の更新): 更新を適用する優先順位。
 - critical (非常事態): この更新を直ちに (14 日以内) 適用することをお勧めします。
 - 重要: ビジネスフローが許可され次第、すぐに (30 日以内) この更新を適用することをお勧めします。
 - medium (中): できるだけ早く (60 日以内) この更新を適用することをお勧めします。
 - low (低): できるだけ早く (90 日以内) この更新を適用することをお勧めします。
 - [エンジンバージョン]: 更新タイプが [エンジン更新] の場合に、更新されるエンジンバージョン。
 - [リリース日]: 更新がリリースされ、クラスターに適用可能になった日。
 - 推奨適用期限: ElastiCache更新を適用するガイダンス日。

- [ステータス]: 更新のステータス。ステータスは以下のとおりです。
 - [利用可能]: 必要なクラスターでこの更新が利用可能です。
 - [完了]: 更新が適用されました。
 - cancelled (キャンセル): 更新はキャンセルされたため、適用する必要はありません。
 - expired (期限切れ): 更新は適用対象外になりました。
4. サービスの更新の詳細を表示するには、(左側のボタンではなく) 個々の更新を選択します。

[Cluster update status] (クラスターの更新ステータス) セクションでは、サービスの更新が適用されていない、または最近適用されたばかりのクラスターのリストを表示できます。クラスターごとに、以下を表示できます。

- クラスター名: クラスターの名前
- ノードを更新しました: 特定のクラスター内で更新された、または特定のサービスの更新に対して利用可能な状態の個々のノードの比率。
- 更新タイプ: サービスの更新のタイプ (セキュリティ更新 または エンジン更新のいずれか)
- ステータス: 更新のステータス。ステータスは以下のとおりです。
 - 使用可能: 必要なクラスターでこの更新が利用可能です。
 - 進行中: このクラスターに更新を適用しています。
 - スケジュール済み: 更新日がスケジュールされています。
 - 完了: 更新が正常に適用されました。完了ステータスのクラスターは、完了後 7 日間表示されます。

ステータスが 使用可能または スケジュール済み (スケジュール済み) であるクラスターのいずれかまたはすべてを選択してから、今すぐ適用を選択した場合、更新がそれらのクラスターに適用され始めます。

AWS CLIを使用してサービスの更新を適用する

サービスの更新が利用可能であるという通知を受け取ったら、AWS CLIを使用してそれらの更新を確認し、適用することができます。

- 利用可能なサービスの更新の説明を取得するには、次のコマンドを実行します。

```
aws elasticache describe-service-updates --service-update-status
available
```

詳細については、を参照してください[describe-service-updates](#)。

- クラスターのリストにサービスの更新を適用するには、次のコマンドを実行します。

```
aws elasticache batch-apply-update-action --service-update
ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
cluster2
```

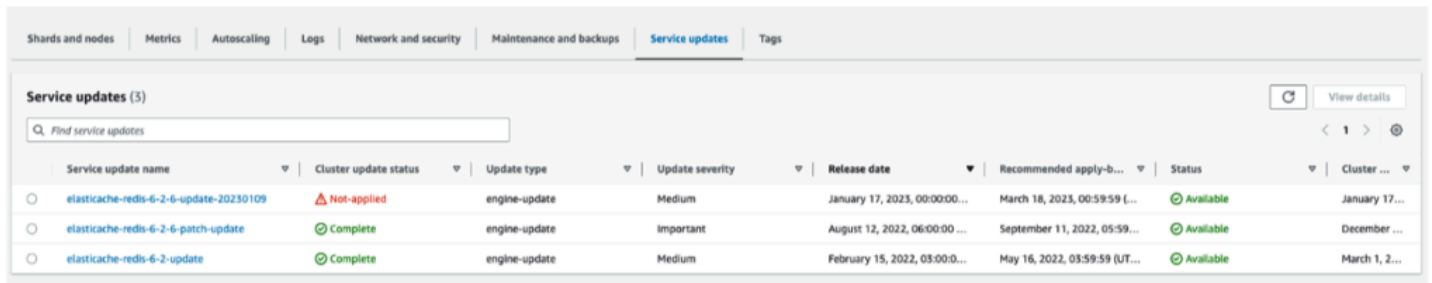
詳細については、を参照してください[batch-apply-update-action](#)。

コンソールを使用して最新のサービスアップデートが適用されていることを確認する AWS

ElastiCache for Redis クラスターが最新のサービスアップデートを実行していることを確認するには、次の手順に従います。

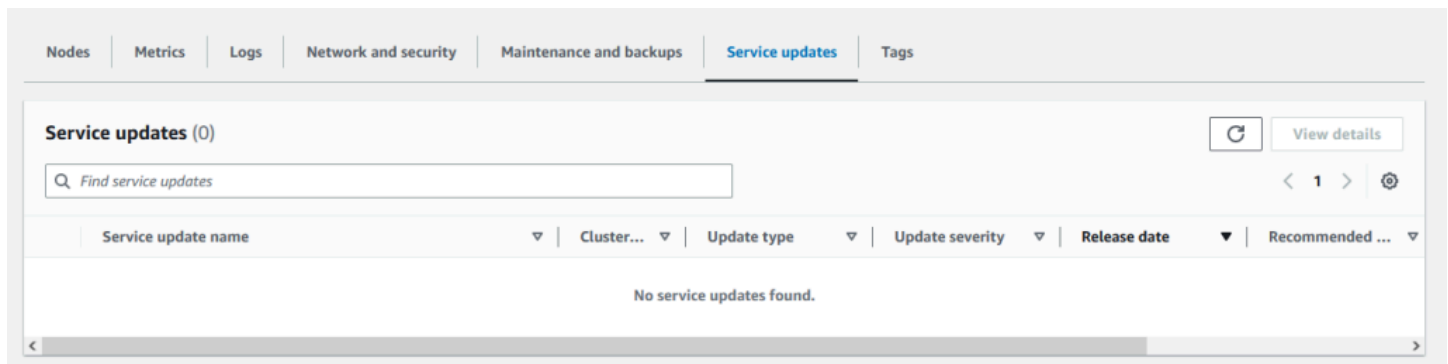
1. Redis クラスターページで該当するクラスターを選択します。
2. ナビゲーションペインで [Service updates] を選択すると、そのクラスターに適用可能なサービスアップデート (ある場合) が表示されます。

コンソールにサービスアップデートのリストが表示されたら、サービスアップデートを選択して [Apply now] を選択できます。



Service update name	Cluster update status	Update type	Update severity	Release date	Recommended apply-b...	Status	Cluster ...
elasticache-redis-6-2-6-update-20230109	Not-applied	engine-update	Medium	January 17, 2023, 00:00:00...	March 18, 2023, 00:59:59 (...)	Available	January 17...
elasticache-redis-6-2-6-patch-update	Complete	engine-update	Important	August 12, 2022, 06:00:00 ...	September 11, 2022, 05:59...	Available	December ...
elasticache-redis-6-2-update	Complete	engine-update	Medium	February 15, 2022, 03:00:0...	May 16, 2022, 05:59:59 (UT...	Available	March 1, 2...

コンソールに「サービスの更新が見つかりません」と表示される場合は、ElastiCache for Redis クラスターに最新のサービス更新がすでに適用されていることを意味します。



サービスの更新の停止

必要に応じて、クラスターの更新を停止できます。たとえば、更新中のクラスターが予期せず急増した場合、更新を停止できます。また、更新に時間がかかりすぎて、ピーク時にビジネスフローを中断する場合は、更新を停止することもできます。

[停止](#)オペレーションでは、そのようなクラスターや、未更新ノードに対する更新はすべて、ただちに中断されます。また、ステータスが [進行中] のノードはすべて [完了] ステータスになります。ただし、ステータスが [更新が利用可能です] の同じクラスター内の他のノードへの更新は中止され、[停止中] ステータスに戻ります。

[停止中] ワークフローが完了すると、ステータスが [停止中] のノードは [停止済み] ステータスに変わります。更新のワークフローによっては、ノードが更新されないクラスターもあります。他のクラスターには、更新されたノードと、ステータスが現在も [更新が利用可能です] のノードが含まれる場合があります。

ビジネスフローを考慮しながら、後に更新プロセスを完了できます。この場合は、更新を完了する適切なクラスターを選択してから、[今すぐ適用] を選択します。詳細については、「[サービスの更新の適用](#)」を参照してください。

コンソールを使用する

ElastiCache コンソールを使用してサービスの更新を中断できます。その方法を以下に示します。

- 選択したクラスターでサービスの更新が進行すると、ElastiCache コンソールのダッシュボード上部に [更新の表示/停止] タブが表示されます。ElastiCache
- 更新を中断するには、[更新を停止する] を選択します。
- 更新を停止したら、クラスターを選択してステータスを確認します。ステータスは [停止中] に戻り、最終的に [停止済み] ステータスになります。

を使用する AWS CLI

サービスの更新は、AWS CLIを使用して中断することができます。次のコード例は、これを実行する方法を説明しています。

レプリケーショングループについては、以下を実行します。

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --replication-group-ids my-replication-group-1 my-replication-group-2
```

キャッシュクラスターでは、以下を実行します。

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --cache-cluster-ids my-cache-cluster-1 my-cache-cluster-2
```

詳細については、[を参照してください](#) [BatchStopUpdateAction](#)。

一般的な脆弱性とリスク (CVE): Redis で対処されたセキュリティの脆弱性 ElastiCache

一般的な脆弱性と露出 (CVE) は、一般的に知られているサイバーセキュリティの脆弱性に関するエントリの一覧です。各エントリは、識別番号、説明、および少なくとも 1 つの公開参照を含むリンクです。このページには、ElastiCache for Redis で対処されたセキュリティ脆弱性のリストが記載されています。

既知の脆弱性から保護するために、ElastiCache 常に最新の Redis バージョンにアップグレードすることをお勧めします。ElastiCache サーバーレスキャッシュを運用すると、CVE 修正がキャッシュに自動的に適用されます。自分で設計したクラスターを操作する場合、ElastiCache for Redis は PATCH コンポーネントを公開します。たとえば、Redis バージョン 6.2.6 ElastiCache で使用する場合、メジャーバージョンは 6、マイナーバージョンは 2、パッチバージョンは 6 です。パッチバージョンは、下位互換性のあるバグ修正、セキュリティ修正、および機能以外の変更を対象としています。

このページを使用して、ElastiCache for Redis の特定のバージョンに特定のセキュリティ脆弱性に対する修正が適用されているかどうかを確認できます。ElastiCache for Redis クラスターがセキュリティ修正のないバージョンを実行している場合は、以下の表を参照して対策を講じてください。修正を含む最新の ElastiCache for Redis バージョンにアップグレードすることも、修正を含む ElastiCache for Redis バージョンを使用している場合は、[を参照して最新のサービスアップデー](#)

トが適用されていることを確認してください。[サービスアップデートの管理](#)サポートされている ElastiCache Redis エンジンバージョンとアップグレード方法の詳細については、[を参照してください](#)。[エンジンバージョンとアップグレード](#)

Note

- CVE が ElastiCache for Redis バージョンで対応されている場合、それは新しいバージョンでも対応されていることを意味します。たとえば、Redis バージョン 6.0.5 ElastiCache で脆弱性に対処した場合、バージョン 6.2.6、7.0.7、7.1 でも引き続き対処されます。
- 次の表のアスタリスク (*) は、セキュリティの脆弱性に対処するために、指定されたバージョンの for Redis を実行している ElastiCache for Redis クラスターに最新のサービスアップデートを適用する必要があることを示しています。ElastiCache クラスターを実行している ElastiCache for Redis バージョンに最新のサービスアップデートが適用されていることを確認する方法の詳細については、[を参照してください](#)。[サービスアップデートの管理](#)

ElastiCache Redis バージョン用	対処済みの CVE
レディス 6.0.5	CVE-2022-24735 * 、 CVE-2022-24736 *
Redis 6.2.6	CVE-2022-24834 * 、 CVE-2022-35977 * 、 CVE-2022-36021 * 、 CVE-2022-24735 、 CVE-2022-24736
Redis 7.0.7	CVE-2023-41056 * 、 CVE-2022-24834 * 、 CVE-2022-35977 、 CVE-2022-36021 、 CVE-2022-24735 、 CVE-2022-24736
Redis 7.1.0	CVE-2023-41056 、 CVE-2022-24834 、 CVE-2022-35977 、 CVE-2022-36021 、 CVE-2022-24735 、 CVE-2022-24736

Amazon ElastiCache でのログ記録とモニタリング

キャッシュを管理するには、キャッシュのパフォーマンスを把握することが重要です。ElastiCache は、キャッシュのパフォーマンスをモニタリングするために Amazon CloudWatch Logs へ発行されるメトリクスを生成します。さらに、ElastiCache は、キャッシュリソースに重大な変更 (新しいキャッシュの作成やキャッシュの削除など) が発生したときにイベントを生成します。

トピック

- [サーバーレスのメトリクスとイベント](#)
- [独自設計型のクラスターメトリクスとイベント](#)
- [AWS CloudTrail を使用した Amazon ElastiCache API コール](#)

サーバーレスのメトリクスとイベント

このセクションでは、サーバーレスキャッシュを使用する際にモニタリングできるメトリクスとイベントを説明しています。

トピック

- [サーバーレスキャッシュメトリクス](#)
- [サーバーレスキャッシュイベント](#)

サーバーレスキャッシュメトリクス

AWS/ElastiCache 名前空間には、Redis サーバーレスキャッシュの次の CloudWatch メトリクスが含まれます。

メトリクス	説明	単位
BytesUsedForCache	キャッシュに保存されているデータによって使用される総バイト数。	バイト
ElastiCacheProcessingUnits	キャッシュ上で実行されたリクエストによって消費された	カウント

メトリクス	説明	単位
	ElastiCacheProcessingUnits (ECPU) の総数。	
SuccessfulReadRequestLatency	読み取りリクエストが成功するまでのレイテンシー。	マイクロ秒
SuccessfulWriteRequestLatency	書き込みリクエストが成功するまでのレイテンシー	マイクロ秒
TotalCmdsCount	キャッシュ上で実行されたすべてのコマンドの合計数。	カウント
CacheHitRate	キャッシュのヒット率を表します。これは、cache_hits と cache_misses 統計を使用して、次の方法で計算されます: $\text{cache_hits} / (\text{cache_hits} + \text{cache_misses})$ 。	割合 (%)
CacheHits	キャッシュで読み取り専用のキー検索に成功した数。	カウント
CurrConnections	キャッシュへのクライアント接続の数。	カウント
ThrottledCmds	ワークロードが ElastiCache のスケーリングよりも速くスケーリングしたため、ElastiCache によってスロットリングされたリクエストの数。	カウント
NewConnections	この期間内にサーバーによって受け入れられた接続の総数。	カウント
CurrItems	キャッシュの項目数。	カウント

メトリクス	説明	単位
CurrVolatileItems	キャッシュ内の TTL を持つ項目の数。	カウント
NetworkBytesIn	キャッシュに転送された合計バイト数	バイト
NetworkBytesOut	キャッシュから転送された合計バイト数	バイト
Evictions	キャッシュによって削除されたキーの数	カウント
IamAuthenticationExpirations	有効期限が切れた IAM で認証された Redis 接続の総数。 IAM を使用した認証 の詳細については、ユーザーガイドで確認できます。	カウント
IamAuthenticationThrottling	スロットリングされた IAM で認証された Redis AUTH または HELLO リクエストの総数。 IAM を使用した認証 の詳細については、ユーザーガイドで確認できます。	カウント
KeyAuthorizationFailures	ユーザーがアクセス許可を持たないキーへのアクセスに失敗した試行の合計数。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント

メトリクス	説明	単位
AuthenticationFailures	AUTH コマンドを使用した Redis への認証に失敗した試行の合計数。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
CommandAuthorizationFailures	ユーザーが呼び出すためのアクセス許可を持たないコマンドの実行に失敗した試行の合計数。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント

コマンドレベルメトリクス

ElastiCache は、以下のコマンドレベルのメトリクスも出力します。ElastiCache は、コマンドタイプごとに、コマンドの合計数と、そのコマンドタイプによって消費される ECPU の数を出力します。

メトリクス	説明	単位
EvalBasedCmds	キャッシュが受信した get コマンドの数。	カウント
EvalBasedCmdsECPUs	eval ベースのコマンドによって消費される ECPU。	カウント
GeoSpatialBasedCmds	地理空間ベースのコマンドの総数。これは、Redis commandstats 統計から算出されます。これは、geo add、geodist、geohash、geopos、georadius	カウント

メトリクス	説明	単位
	、georadius、georadiusbymember など、すべての geo タイプのコマンドを合計して導き出されます。	
GeoSpatialBasedCmdsECPUs	geospatial ベースのコマンドによって消費される ECPU。	カウント
GetTypeCmds	読み取り専用タイプのコマンドの合計数。これは、すべての読み取り専用タイプのコマンド (get、hget、scard、lrange など) を合計することによって Redis commandstats 統計から算出されます。	カウント
GetTypeCmdsECPUs	読み取りコマンドによって消費される ECPU。	カウント
HashBasedCmds	ハッシュベースのコマンドの総数。これは、1 つまたは複数のハッシュに対して実行されるすべてのコマンド (hget、hkeys、hvals、hdel など) を合計することによって Redis commandstats 統計から算出されます。	カウント
HashBasedCmdsECPUs	hash ベースのコマンドによって消費される ECPU。	カウント

メトリクス	説明	単位
HyperLogLogBasedCmds	HyperLogLog ベースのコマンドの合計数。これは、すべての pf タイプのコマンド (pfadd、pfcount、pfmerge など) を合計することによって Redis commandstats 統計から算出されます。	カウント
HyperLogLogBasedCmdsECPUs	HyperLogLog ベースのコマンドによって消費される ECPU。	カウント
JsonBasedCmds	読み取りコマンドと書き込みコマンドの両方を含む JSON コマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON コマンドを合計することで算出されます。	カウント
JsonBasedCmdsECPUs	読み取りコマンドと書き込みコマンドの両方を含む、すべての JSON コマンドによって消費される ECPU。	カウント
JsonBasedGetCmds	JSON 読み取り専用コマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON 読み取りコマンドを合計することで算出されます。	カウント
JsonBasedGetCmdsECPUs	JSON 読み取り専用コマンドによって消費される ECPU。	カウント

メトリクス	説明	単位
JsonBasedSetCmds	JSON 書き込みコマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON 書き込みコマンドを合計することで算出されます。	カウント
JsonBasedSetCmdsECPUs	JSON 書き込みコマンドによって消費される ECPU。	カウント
KeyBasedCmds	キーベースのコマンドの総数。これは、複数のデータ構造で 1 つまたは複数のキーに対して実行されるすべてのコマンド (del、expire、rename など) を合計することによって Redis commandstats 統計から算出されます。	カウント
KeyBasedCmdsECPUs	key ベースのコマンドによって消費される ECPU。	カウント
ListBasedCmds	リストベースのコマンドの総数。これは、1 つまたは複数のリストに対して実行されるすべてのコマンド (lindex、lrange、lpush、ltrim など) を合計することによって Redis commandstats 統計から算出されます。	カウント
ListBasedCmdsECPUs	list ベースのコマンドによって消費される ECPU。	カウント

メトリクス	説明	単位
NonKeyTypeCmds	キーベースではないコマンドの合計数。これは、Redis commandstats 統計に基づき、キーに対して影響を与えないすべてのコマンドを合計することで算出されます (acl、dbsize、info など)。	カウント
NonKeyTypeCmdsECPUs	key ベースでないコマンドによって消費される ECPU。	カウント
PubSubBasedCmds	pub/sub 機能のコマンドの総数。これは、Pub/Sub 機能に使用されるすべてのコマンド (psubscribe、publish、pubsub、punsubscribe、ssubscribe、unsubscribe、spublish、subscribe、unsubscribe) を合計することで、Redis commandstats 統計から算出されます。	カウント
PubSubBasedCmdsECPUs	pub/sub ベースのコマンドによって消費される ECPU。	カウント
SetBasedCmds	セットベースのコマンドの総数。これは、1 つまたは複数のソート対象集合に対して実行されるすべてのコマンド (scard、sdiff、sadd、sunion など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SetBasedCmdsECPUs	set ベースのコマンドによって消費される ECPU。	カウント

メトリクス	説明	単位
SetTypeCmds	書き込みタイプのコマンドの合計数。これは、データを操作するすべての変異型コマンド (set、hset、sadd、lpop など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SetTypeCmdsECPUs	書き込みコマンドによって消費される ECPU。	カウント
SortedSetBasedCmds	ソートされたセットベースのコマンドの総数。これは、1 つまたは複数のソート対象集合に対して実行されるすべてのコマンド (zcount、zrange、zrank、zadd など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SortedSetBasedCmdsECPUs	sorted ベースのコマンドによって消費される ECPU。	カウント
StringBasedCmds	文字列ベースのコマンドの総数。これは、1 つまたは複数の文字列に対して実行されるすべてのコマンド (strlen、setex、setrange など) を合計することによって Redis commandstats 統計から算出されます。	カウント
StringBasedCmdsECPUs	string ベースのコマンドによって消費される ECPU。	カウント

メトリクス	説明	単位
StreamBasedCmds	ストリームベースのコマンドの総数。これは、1つまたは複数のストリームデータのタイプに対して実行されるすべてのコマンド (xrange、xlen、xadd、xdel など) を合計することによって Redis commandstats 統計から算出されます。	カウント
StreamBasedCmdsECPUs	stream ベースのコマンドによって消費される ECPU。	カウント

サーバーレスキャッシュイベント

ElastiCache は、サーバーレスキャッシュに関連するイベントを記録します。この情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などがあります。ElastiCache コンソール、AWS CLI describe-events コマンド、または ElastiCache API アクション DescribeEvents を使用して、ログから簡単にイベントを取得できます。

Amazon EventBridge を使用して ElastiCache イベントのモニタリング、インジェスト、変換、処理できます。詳細については、Amazon EventBridge <https://docs.aws.amazon.com/eventbridge/latest/userguide/> を参照してください。

ElastiCache イベントの表示 (コンソール)

ElastiCache コンソールを使用してイベントを表示するには:

1. AWS Management Consoleにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 利用可能なすべてのイベントのリストを表示するには、ナビゲーションペインで [Events] を選択します。
3. [イベント] 画面では、リストの各行が 1 つのイベントを表し、イベントソース、イベントタイプ、イベントの GMT 時間、イベントの説明が表示されます。Filter を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

ElastiCache イベントの表示 (AWS CLI)

AWS CLI を使用して ElastiCache イベントのリストを作成するには、describe-events コマンドを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、最大 40 個のサーバーレスキャッシュイベントを一覧表示します。

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

次のコードでは、過去 24 時間 (1440 分) のサーバーレスキャッシュのイベントをすべて一覧表示します。

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

サーバーレスイベント

このセクションでは、サーバーレスキャッシュに対して受け取る可能性のあるさまざまなタイプのイベントについて説明します。

サーバーレスキャッシュ作成イベント

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ作成	キャッシュ ARN	作成	サーバーレス キャッシュ	キャッシュ <cache-name> が作成され、使 用できる状態に なりました。
キャッシュ作成	キャッシュ ARN スナップショット ARN	作成	サーバーレス キャッシュ	キャッシュ <cache-name> が作成され、 データがスナッ プショットから 復元されました 。キャッシュが 使用する準備が できました。

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できません でした。VPC エンドポイント を作成するた めの空き IP ア ドレスが不足 しています。
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できませ んでした。リク エストで入力さ れたサブネット が無効です。
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できませ んでした。VPC エンドポイント の作成のクオー タ上限に達しま した。
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できませ んでした。VPC エンドポイント を作成するアク セス許可があり ません。

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できません でした。 互換性のない Redis バージョ ンのユーザーが ユーザーグルー プ <user-group- name> に存在し ます。
キャッシュ作成 失敗	キャッシュ ARN キャッシュス ナップショット ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> を作成できませ んでした。指定 されたユーザー グループ <user- group-name> は 存在しません。

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ作成 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	<p>キャッシュ <cache-name> を作成できませんでした。<reason> という理由により、スナップショットからのデータ復元が失敗しました。</p> <p>失敗の理由:</p> <ul style="list-style-type: none"> • S3 からファイルを取得できませんでした。 • 予想された md5 が実際の md5 と一致しません。 • 提供された RDB ファイルのバージョンはサポートされていません。

サーバーレスキャッシュ更新イベント

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name>

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				用に SecurityGroups が更新されました。
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> のタグが更新 されました。
キャッシュ更新 失敗	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> の更新に失敗 しました。互換 性のない Redis バージョンの ユーザーがユー ザーグループ <user-group- name> に存在し ます。
キャッシュ更新 失敗	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> の更新に失敗し ました。Security Groups 更新に 失敗しました。

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
キャッシュ更新 失敗	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> の更新に失敗 しました。Securit yGroups 更新 は、アクセス許 可が不十分なた め、失敗しまし た。
キャッシュ更新 失敗	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> の更新に失敗し ました。Securit yGroups が無 効なため、Se curityGroups 更 新に失敗しまし た。

サーバーレスキャッシュ削除イベント

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
キャッシュ削除	キャッシュ ARN	削除	サーバーレス キャッシュ	キャッシュ <cache-name> が削除されまし た。

サーバーレスキャッシュ使用量上限イベント

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレス キャッシュ	キャッシュ <cache-name> の上限が更新され ました。
キャッシュの上 限接近	キャッシュ ARN	通知	サーバーレス キャッシュ	スロット <X> が スロットあたり の上限 32 GB の <Y>% 以上を使用 しています。 例: スロット 10 がスロットあたり の上限 32 GB の 90% 以上を使用 しています。
キャッシュ更新 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> が削除されたた め、キャッシュ の上限を更新で きませんでした。 た。
キャッシュ更新 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	キャッシュ <cache-name> の設定が無効 なため、キャッ シユの上限を更 新できませんでした。 した。
キャッシュ更新 失敗	キャッシュ ARN	失敗	サーバーレス キャッシュ	現在キャッシュ されているデー タが新しい制限

詳細タイプ	説明	単位	ソース	メッセージ
				を超えているため、キャッシュ <cache-name> の制限を更新できませんでした。上限を適用する前に、一部のデータをフラッシュしてください。

サーバーレスキャッシュスナップショットイベント

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショット作成	キャッシュ ARN スナップショット ARN	作成	サーバーレス キャッシュス ナップショット	キャッシュ <cache-name> 用にスナップショット <snapshot-name> が作成されました。
スナップショット作成失敗	キャッシュ ARN スナップショット ARN	失敗	サーバーレス キャッシュス ナップショット	キャッシュ <cache-name> のスナップショットを作成できませんでした。<reason>、カスタマー管理キー <key-id> でスナップショット <snapshot-name> を作成

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				<p>できませんでした。</p> <p>失敗理由メッセージ:</p> <ul style="list-style-type: none"> • カスタマー管理キーが無効になっているため • カスタマー管理キーが見つからないため • リクエストがタイムアウトになったため
スナップショット作成失敗	キャッシュ ARN スナップショット ARN	失敗	サーバーレス キャッシュ スナップショット	<p>キャッシュ <cache-name> のスナップショットを作成できませんでした。<reason> という理由により、スナップショット <snapshot-name> の作成に失敗しました。</p> <p>既定の理由:</p> <ul style="list-style-type: none"> • 内部エラーのため

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレス キャッシュス ナップショット	キャッシュ <cache-name> のスナップ ショットをエク スポートできま せませんでした。E lastiCache には バケットへのア クセス許可がな いため、スナッ プショットをバ ケット %s にエ クスポートでき ませんでした。
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレス キャッシュス ナップショット	キャッシュ <cache-name> のスナップ ショットをエク スポートできま せませんでした。バ ケットに同じ名 前のオブジェク トが既に存在す るため、スナッ プショットをバ ケット '%s' にエ クスポートでき ませんでした。

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。バケット所有者のアカウント ID が変更されたため、スナップショットをバケット '%s' にエクスポートできませんでした。
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。S3 バケットにアクセスできないため、スナップショットをバケット '%s' にエクスポートできませんでした。

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。バケットにアクセスできないため、スナップショットをバケット '%s' にエクスポートできませんでした。
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。バケットが存在しないため、スナップショットをバケット '%s' にエクスポートできませんでした。

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。<reason> という理由から、ソーススナップショットのカスタマー管理キー %s でスナップショットをバケット '%s' にエクスポートできませんでした。
スナップショットエクスポート失敗	スナップショット ARN	失敗	サーバーレスキャッシュスナップショット	キャッシュ <cache-name> のスナップショットをエクスポートできませんでした。スナップショットをバケット '%s' にエクスポートできませんでした。

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
スナップショットコピー失敗	スナップショット arn-1 スナップショット arn-2	失敗	サーバーレス キャッシュス ナップショット	スナップショット <snapshot-name> をコピーできませんでした。<reason-name> という理由から、ソーススナップショットのカスタマー管理キー <key-id> を使用してスナップショット '%s' をスナップショット '%s' にコピーできませんでした。
スナップショットコピー失敗	スナップショット arn-1 スナップショット arn-2	失敗	サーバーレス キャッシュス ナップショット	スナップショット <snapshot-name> をコピーできませんでした。ターゲットスナップショットのカスタマー管理キー '%s' '%s' を使用して、スナップショット '%s' をスナップショット '%s' にコピーできませんでした。

独自設計型のクラスターメトリクスとイベント

このセクションでは、独自設計型クラスターを使用する際に予想されるメトリクス、イベント、およびログについて説明します。

トピック

- [独自設計型クラスターのメトリクス](#)
- [独自設計型クラスターのイベント](#)
- [ログ配信](#)
- [CloudWatch メトリクスの使用状況のモニタリング](#)
- [Amazon SNS による ElastiCache イベントのモニタリング](#)

独自設計型クラスターのメトリクス

クラスターを独自に設計する場合は、ElastiCache はホストレベルのメトリクスとキャッシュメトリクスの両方を含むメトリクスを各ノードレベルで出力します。

ホストレベルのメトリクスの詳細については、「[ホストレベルのメトリクス](#)」を参照してください。

ノードレベルのメトリクスの詳細については、「[Redis のメトリクス](#)」を参照してください。

独自設計型クラスターのイベント

ElastiCache は、独自設計型キャッシュに関連するイベントを記録します。独自設計型クラスターを使用する場合、ElastiCache コンソール、AWS CLI、または Amazon Simple Notification Service (SNS) を使用してクラスターイベントを確認できます。独自設計型クラスターのイベントは Amazon EventBridge には公開されません。

独自設計型クラスターのイベント情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などが含まれています。ElastiCache コンソール、AWS CLI describe-events コマンド、または ElastiCache API アクション DescribeEvents を使用して、ログから簡単にイベントを取得できます。

ElastiCache イベントの表示 (コンソール)

次の手順は、ElastiCache コンソールを使用してイベントを表示します。

ElastiCache コンソールを使用してスタックイベントを表示するには

1. AWS Management Consoleにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 利用可能なすべてのイベントのリストを表示するには、ナビゲーションペインで [Events] を選択します。
3. [Events] 画面のリスト内の各行は 1 個のイベントを表し、イベントのソース、イベントの種類、イベントの GMT 時間、イベントの説明が表示されます。Filter を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

ElastiCache イベントの表示 (AWS CLI)

AWS CLI を使用して ElastiCache イベントのリストを作成するには、describe-events コマンドを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、独自設計型クラスターのイベントを最大 40 個一覧表示します。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

次のコードでは、過去 24 時間 (1440 分) の独自設計型クラスターのイベントをすべて一覧表示します。

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

独自設計型クラスターのイベント

このセクションには、独自設計型クラスターで受け取る可能性があるイベントのリストが含まれています。

以下の ElastiCache イベントにより、Amazon SNS 通知がトリガーされます。イベントの詳細については、「[ElastiCache イベントの表示](#)」を参照してください。

イベント名	メッセージ	説明
ElastiCache:AddCacheNodeComplete	ElastiCache:AddCacheNodeComplete : <i>cache-cluster</i>	キャッシュノードがキャッシュクラスターに追加され、使用可能になっています。

イベント名	メッセージ	説明
ElastiCache:AddCacheNodeFailed (使用できる IP アドレスが不足しているため)	ElastiCache:AddCacheNodeFailed : <i>cluster-name</i>	使用できる IP アドレスが不足しているため、キャッシュノードを追加できませんでした。
ElastiCache:CacheClusterParametersChanged	ElastiCache:CacheClusterParametersChanged : <i>cluster-name</i>	1 つ以上のキャッシュクラスターパラメータが変更されました。
ElastiCache:CacheClusterProvisioningComplete	ElastiCache:CacheClusterProvisioningComplete <i>cluster-name-0001-005</i>	キャッシュクラスターのプロビジョニングが完了し、キャッシュクラスター内のキャッシュノードが使用可能になりました。
ElastiCache:CacheClusterProvisioningFailed (ネットワーク状態に互換性がないため)	ElastiCache:CacheClusterProvisioningFailed : <i>cluster-name</i>	存在しない Virtual Private Cloud (VPC) に新しいキャッシュクラスターに起動する試みが行われました。
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : <i>cluster-name</i>	キャッシュクラスターのスケールアップが正常に完了しました。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : <i>cluster-name</i>	キャッシュクラスターのスケールアップが失敗しました。

イベント名	メッセージ	説明
ElastiCache:CacheClusterSecurityGroupModified	ElastiCache:CacheClusterSecurityGroupModified : <i>cluster-name</i>	<p>以下のいずれかのイベントが発生しました。</p> <ul style="list-style-type: none">• キャッシュクラスターに承認されたキャッシュセキュリティグループのリストが修正されました。• 1つ以上の新しい EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループで承認されました。• 1つ以上の EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループから取り消されました。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceStarted	ElastiCache:CacheNodeReplaceStarted : <i>cluster-name</i>	<p>ElastiCache が、キャッシュノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、キャッシュノードの置き換えを開始しました。</p> <div data-bbox="1068 590 1507 905"><p>Note</p><p>置き換えられたキャッシュノードの DNS エントリは変更されません。</p></div> <p>ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のキャッシュクライアントライブラリは、ElastiCache がキャッシュノードを置き換えた後もキャッシュノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。</p>

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceComplete	ElastiCache:CacheNodeReplaceComplete : <i>cluster-name</i>	<p>ElastiCache が、キャッシュノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、キャッシュノードの置き換えを完了しました。</p> <div data-bbox="1068 590 1507 905"><p> Note</p><p>置き換えられたキャッシュノードの DNS エントリは変更されません。</p></div> <p>ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のキャッシュクライアントライブラリは、ElastiCache がキャッシュノードを置き換えた後もキャッシュノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。</p>

イベント名	メッセージ	説明
ElastiCache:CacheNodesRebooted	ElastiCache:CacheNodesRebooted : <i>cluster-name</i>	1つ以上のキャッシュノードが再起動されました。 メッセージ (Memcached): "Cache node %s shutdown" 2番目のメッセージ: "Cache node %s restarted"
ElastiCache:CertificateRenewalComplete (Redis のみ)	ElastiCache:CertificateRenewalComplete	Amazon CA 証明書が正常に更新されました。
ElastiCache:CreateReplicationGroupComplete	ElastiCache:CreateReplicationGroupComplete : <i>cluster-name</i>	レプリケーショングループが正常に作成されました。
ElastiCache>DeleteCacheClusterComplete	ElastiCache>DeleteCacheClusterComplete : <i>cluster-name</i>	キャッシュクラスターと関連するすべてのアプリケーションキャッシュノードの削除が完了しました。
ElastiCache:FailoverComplete (Redis のみ)	ElastiCache:FailoverComplete : <i>mycluster</i>	レプリカノードへのフェイルオーバーが成功しました。
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : <i>cluster-name-0001-005</i>	クラスター内のレプリカの数が増加しました。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : <i>cluster-name-0003-004</i>	クラスターにレプリカを追加するプロセスが開始されました。
ElastiCache:NodeReplacementCanceled	ElastiCache:NodeReplacementCanceled : <i>cluster-name</i>	置き換え対象となっていたクラスター内のノードが置き換え対象ではなくなりました。
ElastiCache:NodeReplacementRescheduled	ElastiCache:NodeReplacementRescheduled : <i>cluster-name</i>	以前置き換え対象になったクラスター内のノードのスケジューリングが、通知に記載されている新しい期間に変更されました。 実行可能なアクションについては、「 ノードの置換 」を参照してください。
ElastiCache:NodeReplacementScheduled	ElastiCache:NodeReplacementScheduled : <i>cluster-name</i>	クラスター内のノードが、通知に記載されている期間中の置き換え対象となりました。 実行可能なアクションについては、「 ノードの置換 」を参照してください。
ElastiCache:RemoveCacheNodeComplete	ElastiCache:RemoveCacheNodeComplete : <i>cluster-name</i>	キャッシュノードがキャッシュクラスターから削除されました。
ElastiCache:ReplicationGroupScalingComplete	ElastiCache:ReplicationGroupScalingComplete : <i>cluster-name</i>	レプリケーショングループのスケールアップオペレーションが正常に完了しました。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupScalingFailed	"Failed applying modification to cache node type to %s."	レプリケーショングループのスケールアップが失敗しました。
ElastiCache:ServiceUpdateAvailableForNode	"Service update is available for cache node %s."	セルフサービス更新は、ノードで使用できます。
ElastiCache:SnapshotComplete (Redis のみ)	ElastiCache:SnapshotComplete : <i>cluster-name</i>	キャッシュスナップショットの作成が正常に完了しました。
ElastiCache:SnapshotFailed (Redis のみ)	SnapshotFailed : <i>cluster-name</i>	キャッシュスナップショットの作成に失敗しました。詳細な原因については、クラスターのキャッシュイベントを参照してください。 スナップショットを表示する場合は、「 DescribeSnapshots 」を参照してください。ステータスは failed です。

ログ配信

Note

Redis スローログは、エンジンバージョン 6.0 以降を使用する Redis キャッシュクラスターおよびレプリケーショングループでサポートされています。

Redis エンジンログは、エンジンバージョン 6.2 以降を使用する Redis キャッシュクラスターおよびレプリケーショングループでサポートされています。

ログ配信により、Redis [SLOWLOG](#) または Redis エンジンログを 2 つの送信先のいずれかにストリーミングできます。

- Amazon Data Firehose
- Amazon CloudWatch Logs

API を使用して ElastiCache APIs。各ログエントリは、次の 2 つの形式のいずれかで、指定された送信先に配信されます: JSON または TEXT。

一定の数のスローログエントリが Redis エンジンから定期的を取得されます。エンジンパラメータ `slowlog-max-len` で指定された値によっては、追加のスローログエントリが送信先に配信されないことがあります。

AWS コンソールまたは変更 APIs の 1 つである [modify-cache-cluster](#) または [modify-replication-group](#) を使用して、配信設定を変更するか、ログ配信を無効にするかを選択できます。

すべてのログ配信の変更では、`apply-immediately` パラメータを設定する必要があります。

Note

Amazon CloudWatch Logs 料金は、ログが Amazon Data Firehose に直接配信された場合でも、ログ配信が有効になっている場合に適用されます。詳細については、[「Amazon CloudWatch 料金表」](#)の「提供されるログ」セクションを参照してください。

スローログエントリの内容

ElastiCache for Redis スローログには、次の情報が含まれています。

- CacheClusterId – キャッシュクラスターの ID
- CacheNodeId – キャッシュノードの ID
- [Id] — スローログエントリごとに一意のプログレッシブ識別子
- [Timestamp] — ログに記録されたコマンドが処理されたときの Unix タイムスタンプ
- [Duration] — 実行に必要な時間 (マイクロ秒単位)
- [Command] — クライアントが使用するコマンド。例えば、set foo barfooはキー、barは値です。Redis ElastiCache の場合、 は実際のキー名と値を に置き換え(2 more arguments)て、機密データが公開されるのを防ぎます。
- ClientAddress – クライアントの IP アドレスとポート
- ClientName — CLIENT SETNAME コマンドで設定されている場合のクライアント名

エンジンログエントリの内容

ElastiCache for Redis エンジンログには、次の情報が含まれています。

- CacheClusterId – キャッシュクラスターの ID
- CacheNodeId – キャッシュノードの ID
- ログレベル — VERBOSE("-"), 、 のいずれか LogLevel を指定できますNOTICE("*")WARNING("#")。
- [Time] (時間) — ログに記録されたメッセージの UTC 時間です。時間の形式は "DD MMM YYYY hh:mm:ss.ms UTC" です。
- [Role] (ロール) — ログの生成元のノードのロールです。「M」(プライマリ) および「S」(レプリカ) および「C」(子プロセス RDB/AOF で使用されているライター) および「X」(センチネル) のいずれかになります。
- [Message] (メッセージ) — Redis エンジンのログメッセージです。

ログ記録を設定するアクセス許可

IAM ユーザー/ロールポリシーに次の IAM アクセス許可を含める必要があります。

- logs:CreateLogDelivery
- logs:UpdateLogDelivery
- logs>DeleteLogDelivery

- logs:GetLogDelivery
- logs>ListLogDeliveries

詳細については、「[アクセス管理の概要: アクセス許可とポリシー](#)」を参照してください。

ログの種類とログ形式の仕様

スローログ

スローログは JSON と TEXT の両方をサポートします

JSON 形式の例を次に示します。

```
{
  "CacheClusterId": "logslowxxxxmsxj",
  "CacheNodeId": "0001",
  "Id": 296,
  "Timestamp": 1605631822,
  "Duration (us)": 0,
  "Command": "GET ... (1 more arguments)",
  "ClientAddress": "192.168.12.104:55452",
  "ClientName": "logslowxxxxmsxj##"
}
```

TEXT 形式の例を次に示します。

```
logslowxxxxmsxj,0001,1605631822,30,GET ... (1 more
arguments),192.168.12.104:55452,logslowxxxxmsxj##
```

エンジンログ

エンジンログは JSON と TEXT の両方をサポートします

JSON 形式の例を次に示します。

```
{
  "CacheClusterId": "xxxxxxxxzy-engine-log-test",
  "CacheNodeId": "0001",
  "LogLevel": "VERBOSE",
  "Role": "M",
}
```

```
"Time": "12 Nov 2020 01:28:57.994 UTC",
"Message": "Replica is waiting for next BGSAVE before synchronizing with the primary.
Check back later"
}
```

TEXT 形式の例を次に示します。

```
xxxxxxxxxxxxzy-engine-log-test/0001:M 29 Oct 2020 20:12:20.499 UTC * A slow-running Lua
script detected that is still in execution after 10000 milliseconds.
```

ElastiCache ログ記録の送信先

このセクションでは、ログ用に選択できる ElastiCache ログ記録の送信先について説明します。各セクションでは、送信先の種類のログを設定するためのガイダンスと、送信先の種類に固有の動作に関する情報を提供します。ログ記録の送信先を設定したら、ログ ElastiCache 記録設定にその仕様を指定して、ログ記録を開始できます。

トピック

- [Amazon CloudWatch Logs](#)
- [Amazon Data Firehose](#)

Amazon CloudWatch Logs

- ログが配信される CloudWatch Logs ロググループを指定します。
- 複数の Redis クラスターおよびレプリケーショングループからのログは、同じロググループに配信できます。
- キャッシュクラスターまたはレプリケーショングループ内の各ノードに対して新しいログストリームが作成され、ログがそれぞれのログストリームに配信されます。ログストリーム名は、次の形式です: `elasticache/${engine-name}/${cache-cluster-id}/${cache-node-id}/${log-type}`

ログを CloudWatch ログに発行するためのアクセス許可

ログを CloudWatch Logs ロググループに送信するように ElastiCache Redis を設定するには、次のアクセス許可設定が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "ElastiCacheLogging"
    },
    {
      "Sid": "ElastiCacheLoggingCWL",
      "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

詳細については、[「ログに送信される CloudWatch ログ」](#)を参照してください。

Amazon Data Firehose

- ログが配信される Firehose 配信ストリームを指定します。
- 複数の Redis クラスターおよびレプリケーショングループからのログは、同じ配信ストリームに配信できます。
- キャッシュクラスターまたはレプリケーショングループ内の各ノードからのログは、同じ配信ストリームに配信されます。各ログメッセージに含まれる `cache-cluster-id` と `cache-node-id` に基づいて、ログメッセージを異なるキャッシュノードから区別できます。

- Firehose へのログ配信は現在、アジアパシフィック (大阪) リージョンではご利用いただけません。

Firehose にログを発行するアクセス許可

Amazon Kinesis Data Firehose 配信ストリームにログを送信するように ElastiCache Redis を設定するには、次のアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "ElastiCacheLogging"
    },
    {
      "Sid": "ElastiCacheLoggingFHSLR",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Sid": "ElastiCacheLoggingFH",
      "Action": [
        "firehose:TagDeliveryStream"
      ],
      "Resource": "Amazon Kinesis Data Firehose delivery stream ARN",
      "Effect": "Allow"
    }
  ]
}
```

```
}
```

コンソールを使用したログ配信の指定

AWS Management Console を使用して、[Redis \(クラスターモードが無効\) クラスターの作成 \(コンソール\)](#) のステップに従って Redis (クラスターモードが無効) クラスターを作成するか、[Redis \(クラスターモードが有効\) クラスターの作成 \(コンソール\)](#) のステップを使用して Redis (クラスターモードが有効) クラスターを作成できます。いずれの場合も、次の手順を実行してログ配信を設定します。

1. [Advanced Redis settings] (Redis の詳細設定) の下で、[Logs] (ログ) を選択してから [Slow logs] (スローログ) または [Engine logs] (エンジンログ) を確認します。
2. [ログの形式] の下で、[テキスト] または [JSON] を選択します。
3. [送信先の種類] の下で、[[CloudWatch Logs] または [Kinesis Firehose] を選択します。
4. [Log destination] (ログの送信先) の下で、[Create new] (新規作成) を選択して Amazon S3 バケット名、CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を入力するか、[Select existing] (既存の選択) を選択してから CloudWatch Logs ロググループ名または Kinesis Data Firehose ストリーム名を選択します。

クラスターの変更時:

ログ配信を有効/無効にするか、送信先の種類、形式、または送信先を変更するかを選択できます。

1. コンソールにサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. ナビゲーションペインで、[Redis clusters] (Redis クラスター) を選択します。
3. クラスターのリストから、変更するクラスターを選択します。横にあるチェックボックスではなく、[クラスター名] を選択します。
4. [クラスター名] ページで、[ログ] タブを選択します。
5. スローログを有効/無効にするには、[スローログを有効にする] または [スローログを無効にする] を選択します。
6. エンジンログを有効/無効にするには、[Enable engine logs] (エンジンログを有効化) または [Disable engine logs] (エンジンログを無効化) のいずれかを選択します。
7. 設定を変更するには、[Modify slow logs] (スローログを変更) または [Modify engine logs] (エンジンログを変更) のいずれかを選択します。

- [送信先の種類] の下で、[[CloudWatch Logs] または [Kinesis Firehose] を選択します。
- [ログの送信先] の下で、[新規作成] を選択するか、CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を入力します。または、[既存のものを選択] を選択してから、CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名を選択します。

を使用したログ配信の指定 AWS CLI

スローログ

CloudWatch ログへのスローログ配信を使用してレプリケーショングループを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id test-slow-log \  
  --replication-group-description test-slow-log \  
  --engine redis \  
  --cache-node-type cache.r5.large \  
  --num-cache-clusters 2 \  
  --log-delivery-configurations '{  
    "LogType":"slow-log",  
    "DestinationType":"cloudwatch-logs",  
    "DestinationDetails":{  
      "CloudWatchLogsDetails":{  
        "LogGroup":"my-log-group"  
      }  
    },  
    "LogFormat":"json"  
  }'
```

Windows の場合:

```
aws elasticache create-replication-group ^  
  --replication-group-id test-slow-log ^  
  --replication-group-description test-slow-log ^  
  --engine redis ^  
  --cache-node-type cache.r5.large ^  
  --num-cache-clusters 2 ^  
  --log-delivery-configurations '{
```



```
"LogType":"slow-log",
"DestinationType":"cloudwatch-logs",
"DestinationDetails":{
  "CloudWatchLogsDetails":{
    "LogGroup":"my-log-group"
  }
},
"LogFormat":"json"
}'
```

スローログを CloudWatch Logs に配信するようにレプリケーショングループを変更する

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \
  --replication-group-id test-slow-log \
  --apply-immediately \
  --log-delivery-configurations '
{
  "LogType":"slow-log",
  "DestinationType":"cloudwatch-logs",
  "DestinationDetails":{
    "CloudWatchLogsDetails":{

      "LogGroup":"my-log-group"
    }
  },
  "LogFormat":"json"
}'
```

Windows の場合:

```
aws elasticache modify-replication-group ^
  --replication-group-id test-slow-log ^
  --apply-immediately ^
  --log-delivery-configurations '
{
  "LogType":"slow-log",
  "DestinationType":"cloudwatch-logs",
  "DestinationDetails":{
    "CloudWatchLogsDetails":{
      "LogGroup":"my-log-group"
    }
  }
}'
```

```
    },  
    "LogFormat": "json"  
  }'  
'
```

スローログ配信を無効にするようにレプリケーショングループを変更します

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id test-slow-log \  
  --apply-immediately \  
  --log-delivery-configurations '  
  {  
    "LogType": "slow-log",  
    "Enabled": false  
  }'  
'
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id test-slow-log ^  
  --apply-immediately ^  
  --log-delivery-configurations '  
  {  
    "LogType": "slow-log",  
    "Enabled": false  
  }'  
'
```

エンジンログ

CloudWatch ログへのエンジンログ配信を使用してレプリケーショングループを作成します。

Linux、macOS、Unix の場合:

```
aws elasticache create-replication-group \  
  --replication-group-id test-slow-log \  
  --replication-group-description test-slow-log \  
  --engine redis \  
  --cache-node-type cache.r5.large \  
  --num-cache-clusters 2 \  
  --log-delivery-configurations '{  
    "LogType": "engine-log",  
    "Enabled": true  
  }'  
'
```

```
"DestinationType":"cloudwatch-logs",
"DestinationDetails":{
  "CloudWatchLogsDetails":{
    "LogGroup":"my-log-group"
  }
},
"LogFormat":"json"
}'
```

Windows の場合:

```
aws elasticache create-replication-group ^
--replication-group-id test-slow-log ^
--replication-group-description test-slow-log ^
--engine redis ^
--cache-node-type cache.r5.large ^
--num-cache-clusters 2 ^
--log-delivery-configurations '{
  "LogType":"engine-log",
  "DestinationType":"cloudwatch-logs",
  "DestinationDetails":{
    "CloudWatchLogsDetails":{
      "LogGroup":"my-log-group"
    }
  },
  "LogFormat":"json"
}'
```

エンジンログを Firehose に配信するようにレプリケーショングループを変更する

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \
--replication-group-id test-slow-log \
--apply-immediately \
--log-delivery-configurations '{
  "LogType":"engine-log",
  "DestinationType":"kinesis-firehose",
  "DestinationDetails":{
    "KinesisFirehoseDetails":{
      "DeliveryStream":"test"
    }
  }
}'
```

```
  },  
  "LogFormat":"json"  
}'
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id test-slow-log ^  
  --apply-immediately ^  
  --log-delivery-configurations '  
  {  
    "LogType":"engine-log",  
    "DestinationType":"kinesis-firehose",  
    "DestinationDetails":{  
      "KinesisFirehoseDetails":{  
        "DeliveryStream":"test"  
      }  
    },  
    "LogFormat":"json"  
  }'
```

エンジン形式に切り替えるようにレプリケーショングループを変更します

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id test-slow-log \  
  --apply-immediately \  
  --log-delivery-configurations '  
  {  
    "LogType":"engine-log",  
    "LogFormat":"json"  
  }'
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id test-slow-log ^  
  --apply-immediately ^  
  --log-delivery-configurations '  
  {  
    "LogType":"engine-log",
```

```
"LogFormat": "json"
}'
```

エンジンログ配信を無効にするようにレプリケーショングループを変更します

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \  
  --replication-group-id test-slow-log \  
  --apply-immediately \  
  --log-delivery-configurations '  
  {  
    "LogType": "engine-log",  
    "Enabled": false  
  }'
```

Windows の場合:

```
aws elasticache modify-replication-group ^  
  --replication-group-id test-slow-log ^  
  --apply-immediately ^  
  --log-delivery-configurations '  
  {  
    "LogType": "engine-log",  
    "Enabled": false  
  }'
```

CloudWatch メトリクスの使用状況のモニタリング

ElastiCache には、クラスターを監視できるようにするメトリクスが用意されています。CloudWatch を通じてこれらのメトリクスにアクセスできます。CloudWatch の詳細については、「[CloudWatch のドキュメント](#)」を参照してください。

ElastiCache では、ホストレベルのメトリクス (たとえば、CPU 使用率など) とキャッシュエンジンソフトウェアに固有のメトリクス (たとえば、キャッシュの取得やキャッシュの損失など) の両方が提供されます。これらのメトリクスは 60 秒間隔で各キャッシュノードに対して測定およびパブリッシュされます。

Important

キャッシュクラスターのパフォーマンスが低下し始めた場合に通知を受け取ることができるよう、特定の主要メトリクスに CloudWatch アラームを設定することを検討してください。詳細については、このガイドの「[モニタリングすべきメトリクス](#)」を参照してください。

トピック

- [ホストレベルのメトリクス](#)
- [Redis のメトリクス](#)
- [モニタリングすべきメトリクス](#)
- [メトリクスの統計と期間の選択](#)
- [CloudWatch クラスターとノードメトリクスのモニタリング](#)

ホストレベルのメトリクス

AWS/ElastiCache 名前空間は、各キャッシュノードに対する以下のホストレベルのメトリクスが含まれます。これらのメトリクスは 60 秒間隔で各キャッシュノードに対して測定およびパブリッシュされます。

以下の資料も参照してください。

- [Redis のメトリクス](#)

メトリクス	説明	単位
CPUUtilization	<p>ホスト全体の CPU 使用率の割合 (%)。Redis はシングルスレッドであるため、4 個以上の vCPU を持つノードで EngineCPU Utilization メトリクスをモニタリングすることをお勧めします。</p>	割合 (%)
CPUCreditBalance	<p>インスタンスが起動または開始後に蓄積した獲得 CPU クレジットの数。T2 スタンドードの場合、CPUCreditBalance には蓄積された起動クレジットの数も含まれます。</p> <p>クレジットは、獲得後にクレジット残高に蓄積され、消費されるとクレジット残高から削除されます。クレジット残高には、インスタンスサイズによって決まる上限があります。制限に到達すると、獲得された新しいクレジットはすべて破棄されます。T2 スタンドードの場合、起動クレジットは制限に対してカウントされません。</p> <p>CPUCreditBalance のクレジットは、インスタンスがそのベースライン CPU 使用率を超えてバーストするために消費できます。</p> <p>CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。</p> <p>このメトリクスは、T2 バーストパフォーマンスインスタンスでは利用できません。</p>	クレジット (vCPU 分)
CPUCreditUsage	<p>CPU 使用率に関してインスタンスで消費される CPU クレジットの数。1 つの CPU クレジットは、1 個の vCPU が 100% の使用率で 1 分間実行されること、または、vCPU、使用率、時間の同等の組み合わせ (例えば、1 個の</p>	クレジット (vCPU 分)

メトリクス	説明	単位
	<p>vCPU が 50% の使用率で 2 分間実行されるか、2 個の vCPU が 25% の使用率で 2 分間実行される) に相当します。</p> <p>CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。5 分を超える期間を指定する場合は、Average 統計の代わりに Sum 統計を使用します。</p> <p>このメトリクスは、T2 バーストパフォーマンスインスタンスでは利用できません。</p>	
FreeableMemory	<p>ホストで使用可能な空きメモリの量。OS によって解放できる可能性があるとしてレポートされる RAM、バッファ、およびキャッシュから算出されます。</p>	バイト
NetworkBytesIn	<p>ホストがネットワークから読み取ったバイト数。</p>	バイト
NetworkBytesOut	<p>すべてのネットワークインターフェイスでの、このインスタンスから送信されたバイトの数。</p>	バイト
NetworkPacketsIn	<p>すべてのネットワークインターフェイスでの、このインスタンスによって受信されたパケットの数。このメトリクスは、受信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。</p>	カウント
NetworkPacketsOut	<p>すべてのネットワークインターフェイスでの、このインスタンスから送信されたパケットの数。このメトリクスは、送信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。</p>	カウント

メトリクス	説明	単位
NetworkBandwidthInAllowanceExceeded	インバウンド集計帯域幅がインスタンスの最大値を超えたためにキューまたはドロップされたパケットの数。	カウント
NetworkConntrackAllowanceExceeded	接続トラッキングがインスタンスの最大数を超え、新しい接続を確立できなかったためにドロップされたパケットの数。これにより、インスタンスとの間で送受信されるトラフィックのパケット損失が発生する可能性があります。	カウント
NetworkBandwidthOutAllowanceExceeded	アウトバウンド集計帯域幅がインスタンスの最大値を超えたためにキューまたはドロップされたパケットの数。	カウント
NetworkPacketsPerSecondAllowanceExceeded	1秒あたりの双方向パケットがインスタンスの最大値を超えたためにキューまたはドロップされたパケットの数。	カウント
NetworkMaxBytesIn	1分あたりの受信バイトの最大バースト。	バイト
NetworkMaxBytesOut	1分あたりの送信バイトの最大バースト。	バイト
NetworkMaxPacketsIn	1分あたりの受信パケットの最大バースト。	カウント
NetworkMaxPacketsOut	1分あたりの送信パケットの最大バースト。	カウント
SwapUsage	ホストで使用されるスワップの量。	バイト

Redis のメトリクス

AWS/ElastiCache 名前空間には、次の Redis メトリクスが含まれます。

ReplicationLag および EngineCPUUtilization を除き、これらのメトリクスは、Redis の info コマンドから算出されます。各メトリクスは、キャッシュノードレベルで算出されます。

Redis の info コマンドの詳細は、「<http://redis.io/commands/info>」を参照してください。

以下の資料も参照してください。

• ホストレベルのメトリクス

メトリクス	説明	単位
ActiveDefragHits	アクティブなデフラグメンテーションプロセスで実行された 1 分あたりの値の再割り当て数。これは、 Redis INFO での active_defrag_hits 統計から算出されます。	数
AuthenticationFailures	AUTH コマンドを使用した Redis への認証に失敗した試行の合計数。個々の認証失敗の詳細については、 ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
BytesUsedForCache	データセット、バッファなど、すべての目的で Redis によって割り当てられた合計バイト数。	バイト
	データ階層化 を使用する Redis クラスターの Dimension: Tier=Memory : メモリによってキャッシュに使用される合計バイト数です。これは、 Redis INFO での used_memory 統計の値です。	バイト
	データ階層化 を使用する Redis クラスターの Dimension: Tier=SSD : SSD によってキャッシュに使用される合計バイト数です。	バイト
BytesReadFromDisk	ディスクから読み取られる 1 分あたりの合計バイト数です。 データ階層化 を使用するクラスターのみがサポートされます。	バイト
BytesWrittenToDisk	ディスクに書き込まれる 1 分あたりの合計バイト数です。 データ階層化 を使用するクラスターのみがサポートされます。	バイト


メトリクス	説明	単位
CacheHits	メインディクショナリで読み取り専用のキー検索に成功した数。これは、 Redis INFO での <code>keyspace_hits</code> 統計から算出されます。	カウント
CacheMisses	メインディクショナリで読み取り専用のキー検索に失敗した数。これは、 Redis INFO での <code>keyspace_misses</code> 統計から算出されます。	カウント
CommandAuthorizationFailures	ユーザーが呼び出すためのアクセス許可を持たないコマンドの実行に失敗した試行の合計数。個々の認証失敗の詳細については、 ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
CacheHitRate	Redis インスタンスの使用効率を示します。キャッシュ比率が約 0.8 より小さい場合、かなりの量のキーが削除された、期限切れになった、または存在しないことを意味します。これは、 <code>cache_hits</code> と <code>cache_misses</code> 統計を使用して、次の方法で計算されます: $\text{cache_hits} / (\text{cache_hits} + \text{cache_misses})$ 。	割合 (%)
ChannelAuthorizationFailures	ユーザーがアクセス許可を持たないチャンネルへのアクセスに失敗した試行の合計数。個々の認証失敗の詳細については、 ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このメトリクスにアラームを設定することをお勧めします。	カウント

メトリクス	説明	単位
CurrConnections	リードレプリカからの接続を除くクライアント接続の数。各ケースでクラスターをモニタリングするために 2~4 個の接続 ElastiCache を使用します。これは、 Redis INFO での <code>connected_clients</code> 統計から算出されます。	カウント
CurrItems	キャッシュの項目数。これは、Redis <code>keyspace</code> 統計に基づき、キー空間全体のすべてのキーを合計することで算出されます。	カウント
	データ階層化 を使用するクラスターの <code>Dimension: Tier=Memory</code> です。メモリ内の項目の数です。	カウント
	データ階層化 を使用するクラスターの <code>Dimension: Tier=SSD</code> (ソリッドステートドライブ) です。SSD 内の項目の数です。	カウント
CurrVolatileItems	TTL が設定されているすべてのデータベース内のキーの総数。これは、Redis <code>expires</code> 統計に基づき、キー空間全体で TTL 設定を持つすべてのキーを合計することで算出されます。	カウント
DatabaseCapacityUsagePercentage	<p>使用中のクラスターの総データ容量の割合。</p> <p>データ階層型インスタンスでは、メトリクスはとして計算され $(used_memory - mem_not_counted_for_evict + SSD\ used) / (maxmemory + SSD\ total\ capacity)$、<code>used_memory</code> と <code>maxmemory</code> は Redis INFO から取得されます。</p> <p>それ以外の場合、メトリクスはを使用して計算されます $used_memory / maxmemory$。</p>	割合 (%)

メトリクス	説明	単位
DatabaseCapacityUsageCountedForEvictPercentage	<p>オーバーヘッドと COB に使用される総データ容量を除く、使用中のクラスターのメモリの割合です。このメトリクスは次のように計算されます。</p> $\frac{\text{used_memory} - \text{mem_not_counted_for_evict}}{\text{maxmemory}}$ <p>データ階層化インスタンスでは、メトリクスは次のように計算されます。</p> $\frac{(\text{used_memory} + \text{SSD used})}{(\text{maxmemory} + \text{SSD total capacity})}$ <p>used_memory と maxmemory は Redis INFO から取得したものです。</p>	割合 (%)
DatabaseMemoryUsagePercentage	<p>使用中のクラスターのメモリの割合。これは、Redis INFO からの used_memory/maxmemory を使用して計算されます。</p>	割合 (%)
DatabaseMemoryUsageCountedForEvictPercentage	<p>オーバーヘッドと COB に使用されるメモリを除く、使用中のクラスターのメモリの割合です。これは、Redis INFO からの used_memory-mem_not_counted_for_evict/maxmemory を使用して計算されます。</p>	割合 (%)

メトリクス	説明	単位
DB0AverageTTL	<p>Redis INFO コマンドの <code>keyspace</code> 統計からの <code>DB0</code> の <code>avg_ttl</code> を公開します。レプリカはキーを失効させず、プライマリノードがキーを失効させるまで待機します。プライマリノードがキーを失効させる (または LRU のためにキーを削除する) と、プライマリノードは <code>DEL</code> コマンドを合成し、それはすべてのレプリカに送信されます。したがって、レプリカノードではキーの有効期限がないため、<code>DB0AverageTTL</code> は 0 になり、TTL を追跡しません。</p>	ミリ秒

メトリクス	説明	単位
EngineCPUUtilization	<p>Redis エンジンスレッドの CPU 使用率を提供します。Redis はシングルスレッドであるため、このメトリクスを使用して、Redis プロセス自体のロードを分析できます。</p> <p>EngineCPUUtilization メトリクスは、Redis プロセスのより正確な可視性を提供します。CPUUtilization メトリクスと組み合わせてそれを使用できます。CPUUtilization は、他のオペレーティングシステムや管理プロセスを含むサーバーインスタンス全体の CPU 使用率を公開します。4 個以上の vCPU を持つ大きなノードの場合は、EngineCPUUtilization メトリクスを使用して、スケーリングのしきい値をモニタリングおよび設定します。</p>	割合 (%)

 Note

ElastiCache ホストでは、バックグラウンドプロセスがホストをモニタリングして、マネージドデータベースエクスペリエンスを提供します。これらのバックグラウンドプロセスは、CPU ワークロードのかなりの部分を占有する可能性があります。これは、vCPU が 2 個を超える大規模なホストでは重要ではありません。ただし、vCPU が 2 個以下の小規模なホストには影響を与える可能性があります。EngineCPUUtilization メトリクスのみをモニタリングする場合、Redis からの CPU 使用率と、バックグラウンドモニタリングプロセスからの CPU 使用率の両方が高く、ホストが過負荷になっている状況には気付くことができ

メトリクス	説明	単位
	<p>ません。したがって、vCPU が 2 個以下のホストについては、CPUUtilization メトリクスをモニタリングすることをお勧めします。</p>	
Evictions	<p>maxmemory の制限のため排除されたキーの数。これは、Redis INFO での evicted_keys 統計から算出されます。</p>	カウント
GlobalDatastoreReplicationLag	<p>これは、セカンダリリージョンのプライマリノードとプライマリリージョンのプライマリノード間の遅延です。クラスターモードが有効な Redis の場合、遅延はシャード間の最大遅延を示します。</p>	[秒]
IamAuthenticationExpirations	<p>有効期限が切れた IAM で認証された Redis 接続の総数。IAM を使用した認証 の詳細については、ユーザーガイドで確認できます。</p>	カウント
IamAuthenticationThrottling	<p>スロットリングされた IAM で認証された Redis AUTH または HELLO リクエストの総数。IAM を使用した認証 の詳細については、ユーザーガイドで確認できます。</p>	カウント
IsMaster	<p>ノードが現在のシャード/クラスターのプライマリノードかどうかを示します。メトリクスは 0 (プライマリではない) または 1 (プライマリ) にすることができます。</p>	カウント

メトリクス	説明	単位
KeyAuthorizationFailures	ユーザーがアクセス許可を持たないキーへのアクセスに失敗した試行の合計数。個々の認証失敗の詳細については、 ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
KeysTracked	Redis のキートラッキングによって追跡されるキーの数 (tracking-table-max-keys のパーセンテージ)。キーラッキングは、クライアント側のキャッシュを支援するために使用され、キーが変更されたときにクライアントに通知します。	カウント
MemoryFragmentationRatio	Redis エンジンのメモリ割り当ての効率を示します。特定のしきい値は、異なる動作を意味します。推奨値は、1.0 を超える断片化です。これは、 Redis INFO の mem_fragmentation_ratio statistic から計算されます。	数

メトリクス	説明	単位
NewConnections	<p>この期間内にサーバーによって受け入れられた接続の総数。これは、Redis INFO での <code>total_connections_received</code> 統計から算出されます。</p> <div data-bbox="594 447 1268 1045" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>Redis バージョン 5 以前 ElastiCache で使用している場合、このメトリクスによって報告される 2 つから 4 つの接続が ElastiCache によってクラスターのモニタリングに使用されます。ただし、for Redis ElastiCache バージョン 6 以降で使用する場合、クラスターのモニタリング ElastiCache に使用する接続は、このメトリクスに含まれません。</p> </div>	カウント
NumItemsReadFromDisk	ディスクから取得される 1 分あたりの項目の総数です。 データ階層化 を使用するクラスターのみがサポートされます。	カウント
NumItemsWrittenToDisk	ディスクに書き込まれる 1 分あたりの項目の総数です。 データ階層化 を使用するクラスターのみがサポートされます。	カウント
MasterLinkHealthStatus	このステータスの値は、0 または 1 のいずれかになります。値 0 は、プライマリノードの ElastiCache データが EC2 の Redis と同期していないことを示します。値 1 は、データが同期されていることを示します。移行を完了するには、 CompleteMigration API オペレーションを使用します。	ブール値

メトリクス	説明	単位
Reclaimed	キーの有効期限切れイベントの総数。これは、 Redis INFO での <code>expired_keys</code> 統計から算出されます。	カウント
ReplicationBytes	レプリケートされたノードについては、 <code>ReplicationBytes</code> は、プライマリがすべてのレプリカに対して送信するバイト数を報告します。このメトリクスは、レプリケーショングループに対する書き込み負荷を表します。これは、 [Redis INFO] での <code>master_repl_offset</code> 統計から算出されます。	バイト
ReplicationLag	このメトリクスは、リードレプリカとして実行中のノードにのみ適用できます。レプリカのプライマリノードからの変更適用の進行状況を秒で表します。Redis エンジンバージョン 5.0.6 以降では、ラグはミリ秒単位で測定できます。	[秒]
SaveInProgress	このバイナリメトリクスは、バックグラウンド保存 (分岐または分岐なし) が進行中の場合は常に 1 を返し、それ以外の場合は 0 を返します。バックグラウンド保存プロセスは一般に、スナップショットおよび同期の際に使用されます。これらのオペレーションによりパフォーマンスが低下する可能性があります。SaveInProgress メトリクスを使用して、パフォーマンスが低下した原因がバックグラウンド保存プロセスであるかどうかを診断できます。これは、 [Redis INFO] での <code>rdb_bgsave_in_progress</code> 統計から算出されます。	ブール値

メトリクス	説明	単位
TrafficManagementActive	<p>for ElastiCache Redis が、受信コマンド、モニタリング、またはレプリケーションに割り当てられたトラフィックを調整することで、トラフィックをアクティブに管理しているかどうかを示します。トラフィックは、Redis が処理できる数よりも多くのコマンドがノードに送信された場合に管理され、エンジンの安定性と最適な動作を維持するために使用されます。データポイントが 1 の場合は、提供されているワークロードに対してノードが過小評価されていることを示している可能性があります。</p> <div style="border: 1px solid #00a0e3; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>この指標が引き続き有効な場合は、クラスターを評価してスケールアップとスケールアウトのどちらが必要かを判断します。関連するメトリクスには、NetworkBandwidthOutAllowanceExceeded および EngineCPUUtilization が含まれます。</p> </div>	ブール値

EngineCPUUtilization の利用可能性

AWS 以下にリストされているリージョンは、サポートされているすべてのノードタイプで使用できます。

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

リージョン	リージョン名
us-west-1	米国西部 (北カリフォルニア)
us-west-2	米国西部 (オレゴン)
ap-northeast-1	アジアパシフィック (東京)
ap-northeast-2	アジアパシフィック (ソウル)
ap-northeast-3	アジアパシフィック (大阪)
ap-east-1	アジアパシフィック (香港)
ap-south-1	アジアパシフィック (ムンバイ)
ap-southeast-1	アジアパシフィック (シンガポール)
ap-southeast-2	アジアパシフィック (シドニー)
ap-southeast-3	アジアパシフィック (ジャカルタ)
ca-central-1	カナダ (中部)
cn-north-1	中国 (北京)
cn-northwest-2	中国 (寧夏)
me-south-1	中東 (バーレーン)
eu-central-1	欧州 (フランクフルト)
eu-west-1	欧州 (アイルランド)
eu-west-2	欧州 (ロンドン)
eu-west-3	欧州 (パリ)
eu-south-1	欧州 (ミラノ)
af-south-1	アフリカ (ケープタウン)

リージョン	リージョン名
eu-north-1	欧州 (ストックホルム)
sa-east-1	南米 (サンパウロ)
us-gov-west-1	AWS GovCloud (米国西部)
us-gov-east-1	AWS GovCloud (米国東部)

以下は特定の種類のコマンドの集計で、`info commandstats` から算出されています。commandstats セクションには、コール数、これらのコマンドによって消費された合計 CPU 時間、およびコマンド実行あたりの平均 CPU 消費など、コマンドタイプに基づいた統計情報が表示されます。コマンドタイプごとに、次の行が追加されます: `cmdstat_XXX: calls=XXX, usec=XXX, usec_per_call=XXX`。

以下に示すレイテンシーメトリクスは、[Redis INFO](#) からの commandstats 統計を使用して計算されます。それらは次のように計算されます: $\text{delta}(\text{usec})/\text{delta}(\text{calls})$ 。delta は、1 分以内の差分として計算されます。レイテンシーは、がコマンドを処理する ElastiCache のにかかる CPU 時間として定義されます。データ階層化を使用するクラスターの場合、SSD から項目を取得するのにかかる時間はこれらの測定に含まれないことにご注意ください。

利用可能なコマンドの完全なリストについては、Redis ドキュメントの「[Redis コマンド](#)」を参照してください。

メトリクス	説明	単位
ClusterBasedCmds	クラスターベースのコマンドの総数。これは、クラスターに対して実行されるすべてのコマンド (<code>cluster slot</code> 、 <code>cluster info</code> など) を合計することによって Redis commandstats 統計から算出されます。	カウント
ClusterBasedCmdsLatency	クラスターベースのコマンドのレイテンシー。	マイクロ秒
EvalBasedCmds	eval ベースのコマンドの合計数。これは、eval、evalsha を合計することによって	カウント

メトリクス	説明	単位
	Redis commandstats 統計から算出されま す。	
EvalBasedCmdsLatency	Eval ベースのコマンドのレイテンシー。	マイクロ秒
GeoSpatialBasedCmds	地理空間ベースのコマンドの総数。これは Redis commandstats 統計から算出されま す。これは、すべての geo の種類のコマンド (geoadd、geodist、geohash、geopos、georadiu 、および georadiusbymember) を合計すること によって算出されます。	カウント
GeoSpatialBasedCmd sLatency	地理空間ベースのコマンドのレイテンシー。	マイクロ秒
GetTypeCmds	read-only 型のコマンドの合計数。これ は、すべての read-only の種類のコマンド (get、hget、scard、lrange など) を合計するこ とによって Redis commandstats 統計から算 出されます。	カウント
GetTypeCmdsLatency	読み取りコマンドのレイテンシー。	マイクロ秒
HashBasedCmds	ハッシュベースのコマンドの総数。これは、1 つ以上のハッシュに対して実行されるすべての コマンド (hget、hkeys、hvals、hdel など) を 合計することによって Redis commandstats 統計から算出されます。	カウント
HashBasedCm dsLatency	ハッシュベースのコマンドのレイテンシー。	マイクロ秒
HyperLogLogBasedCmds	HyperLogLog ベースのコマンドの合計 数。これは、すべての pf の種類のコマンド (pfadd、pfcount、pfmerge など) を合計するこ とによって Redis commandstats 統計から算 出されます。	カウント

メトリクス	説明	単位
HyperLogLogBasedCmdsLatency	HyperLogLogベースのコマンドのレイテンシー。	マイクロ秒
JsonBasedCmds	読み取りコマンドと書き込みコマンドの両方を含む JSON コマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON コマンドを合計することで算出されます。	カウント
JsonBasedCmdsLatency	読み取りコマンドと書き込みコマンドの両方を含む、すべての JSON コマンドのレイテンシー。	マイクロ秒
JsonBasedGetCmds	JSON 読み取り専用コマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON 読み取りコマンドを合計することで算出されます。	カウント
JsonBasedGetCmdsレイテンシー	JSON 読み取り専用コマンドのレイテンシー。	マイクロ秒
JsonBasedSetCmds	JSON 書き込みコマンドの合計数。これは、Redis commandstats 統計に基づき、JSON キーに影響を与えるすべての JSON 書き込みコマンドを合計することで算出されます。	カウント
JsonBasedSetCmdsレイテンシー	JSON 書き込みコマンドのレイテンシー。	マイクロ秒
KeyBasedCmds	キーベースのコマンドの総数。これは、複数のデータ構造で 1 つ以上のキーに対して実行されるすべてのコマンド (del、expire、rename など) を合計することによって Redis commandstats 統計から算出されます。	カウント

メトリクス	説明	単位
KeyBasedCmdsLatency	キーベースのコマンドのレイテンシー。	マイクロ秒
ListBasedCmds	リストベースのコマンドの総数。これは、1つ以上のリストに対して実行されるすべてのコマンド (lindex、lrange、lpush、ltrim など) を合計することによって Redis commandstats 統計から算出されます。	カウント
ListBasedCmdsLatency	リストベースのコマンドのレイテンシー。	マイクロ秒
NonKeyTypeCmds	キーベースではないコマンドの合計数。これは、Redis commandstats 統計に基づき、キーに対して影響を与えないすべてのコマンドを合計することで算出されます (acl、dbsize、info など)。	カウント
NonKeyTypeCmdsレイテンシー	non-key-based コマンドのレイテンシー。	マイクロ秒
PubSubBasedCmds	pub/sub 機能のコマンドの総数。これは、pub/sub 機能で使用されるすべてのコマンド (psubscribe、publish、pubsub、punsubscribe、ssubscribe、sunsubscribe、spublish、subscribe、unsubscribe) を合計することによって Redis commandstats 統計から算出されます。	カウント
PubSubBasedCmdsLatency	PubSubBased コマンドのレイテンシー。	マイクロ秒

メトリクス	説明	単位
SetBasedCmds	セットベースのコマンドの総数。これは、1つ以上のソートされたセットに対して実行されるすべてのコマンド (scard、sdiff、sadd、sunion など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SetBasedCmdsLatency	セットベースのコマンドのレイテンシー。	マイクロ秒
SetTypeCmds	write 型のコマンドの合計数。これは、データ上で動作する mutative の種類のすべてのコマンド (set、hset、sadd、lpop など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SetTypeCmdsLatency	書き込みコマンドのレイテンシー。	マイクロ秒
SortedSetBasedCmds	ソートされたセットベースのコマンドの総数。これは、1つ以上のソートされたセットに対して実行されるすべてのコマンド (zcount、zrange、zrank、zadd など) を合計することによって Redis commandstats 統計から算出されます。	カウント
SortedSetBasedCmdsLatency	ソートベースのコマンドのレイテンシー。	マイクロ秒
StringBasedCmds	文字列ベースのコマンドの総数。これは、1つ以上の文字列に対して実行されるすべてのコマンド (strlen、setex、setrange など) を合計することによって Redis commandstats 統計から算出されます。	カウント
StringBasedCmdsLatency	文字列ベースのコマンドのレイテンシー。	マイクロ秒

メトリクス	説明	単位
StreamBasedCmds	ストリームベースのコマンドの総数。これは、1つ以上のストリームデータの種別に対して実行されるすべてのコマンド (xrange、xlen、xadd、xdel など) を合計することによって Redis commandstats 統計から算出されます。	カウント
StreamBasedCmdsLatency	ストリームベースのコマンドのレイテンシー。	マイクロ秒

モニタリングすべきメトリクス

次の CloudWatch メトリクスは、ElastiCache パフォーマンスを把握するのに役立ちます。ほとんどの場合、パフォーマンスの問題が発生する前に修正作業を行うことができるように、これらのメトリクスに CloudWatch アラームを設定することをお勧めします。

モニタリングするメトリクス

- [CPUUtilization](#)
- [EngineCPUUtilization](#)
- [SwapUsage](#)
- [Evictions](#)
- [CurrConnections](#)
- [メモリ](#)
- [ネットワーク](#)
- [レイテンシー](#)
- [レプリケーション](#)
- [トラフィック管理](#)

CPUUtilization

パーセント値でレポートされるホストレベルのメトリクスです。詳細については、「[ホストレベルのメトリクス](#)」を参照してください。

2 個以下の vCPU を持つ小さなノードタイプの場合は、CPUUtilization メトリクスを使用してワークロードをモニタリングします。

一般的に、利用可能な CPU の 90% にしきい値を設定することをお勧めします。Redis はシングルスレッドであるため、実際のしきい値はノードの総容量の割合として計算します。たとえば、2 個のコアを搭載するノードタイプを使用しているとします。この場合、CPUUtilization のしきい値は $90/2$ 、つまり 45% になります。

使用しているキャッシュノードのコア数に基づいて独自のしきい値を決定する必要があります。このしきい値を超えた場合で、主なワークロードが読み込みリクエストから生成されている場合、リードレプリカを追加してキャッシュクラスターをスケールします。主なワークロードが書き込みリクエストからのものである場合、クラスター設定に応じて、以下のことをお勧めします。

- Redis (クラスターモードが無効) クラスター: より大きなキャッシュインスタンスタイプを使用してスケールアップします。
- Redis (クラスターモードが有効) クラスター: より多くのシャードを追加して、より多くのプライマリノード間で書き込みワークロードを分散します。

Tip

Redis ユーザーは、ホストレベルのメトリクス CPUUtilization を使用する代わりに、Redis エンジンコアでの使用量のパーセント値をレポートする Redis メトリクス EngineCPUUtilization を使用できる場合があります。コードでこのメトリクスが利用できるかどうか、およびその詳細については、「[Redis のメトリクス](#)」を参照してください。

4 個以上の vCPU を持つ大きなノードタイプでは、Redis エンジンコアでの使用量のパーセント値をレポートする EngineCPUUtilization メトリクスを使用することをお勧めします。コードでこのメトリクスが利用できるかどうか、およびその詳細については、「[Redis のメトリクス](#)」を参照してください。

EngineCPUUtilization

4 個以上の vCPU を持つ大きなノードタイプでは、Redis エンジンコアでの使用量のパーセント値をレポートする EngineCPUUtilization メトリクスを使用することをお勧めします。コードでこのメトリクスが利用できるかどうか、およびその詳細については、「[Redis のメトリクス](#)」を参照してください。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「CPU」セクションを参照してください。

SwapUsage

バイト単位でレポートされるホストレベルのメトリクスです。詳細については、「[ホストレベルのメトリクス](#)」を参照してください。

FreeableMemory CloudWatch メトリクスが 0 に近い (つまり 100 MB 未満) または SwapUsage メトリクスが FreeableMemory メトリクスより大きい場合、ノードがメモリプレッシャーを受けていることを示します。このような場合は、以下のトピックを参照してください。

- [Redis スナップショットを作成するのに十分なメモリがあることの確認](#)
- [予約メモリの管理](#)

Evictions

これは、キャッシュエンジンのメトリクスです。アプリケーションニーズに基づいてこのメトリクスの独自のアラームしきい値を決定することをお勧めします。

CurrConnections

これは、キャッシュエンジンのメトリクスです。アプリケーションニーズに基づいてこのメトリクスの独自のアラームしきい値を決定することをお勧めします。

CurrConnections の値が大きくなった場合、アプリケーションに問題があることを示している可能性があります。アプリケーション動作を調査してこの問題を解決する必要があります。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「接続」セクションを参照してください。

メモリ

メモリは Redis の中核的な側面です。クラスターのメモリ使用率を理解することは、データの損失を回避し、データセットの将来の増加に対応するために必要です。ノードのメモリ使用率に関する統計は、Redis [INFO](#) コマンドのメモリセクションで利用できます。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「メモリ」セクションを参照してください。

ネットワーク

クラスターのネットワーク帯域幅容量の決定要因の 1 つは、選択したノードの種類です。ノードのネットワーク容量の詳細については、「[Amazon ElastiCache 料金表](#)」を参照してください。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「ネットワーク」セクションを参照してください。

レイテンシー

データ構造ごとに集約されたレイテンシーを提供する CloudWatch メトリクスのセットを使用して、コマンドのレイテンシーを測定できます。これらのレイテンシーメトリクスは、commandstatsRedis [INFO](#) から計を使用して計算されます。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「レイテンシー」セクションを参照してください。

レプリケーション

レプリケーションされるデータの量は、ReplicationBytes メトリクスを介して見ることができます。このメトリクスは、レプリケーショングループに対する書き込み負荷を表しますが、レプリケーションの状態に関するインサイトは提供されません。この目的のために、ReplicationLag メトリクスを使用できます。

詳細については、[Amazon CloudWatch を使用した Amazon ElastiCache for Redis によるベストプラクティスのモニタリング](#)の「レプリケーション」セクションを参照してください。

トラフィック管理

ElastiCache for Redis は、処理できる数よりも多くのコマンドがノードに送信された場合に、ノードに対するトラフィックを自動的に管理します。これにより、エンジンの最適な動作と安定性を維持します。

ノードでトラフィックがアクティブに管理されている場合、メトリクス TrafficManagementActive は 1 のデータポイントを出力します。これは、提供されているワークロードに対してノードが過小評価されている可能性を示します。このメトリクスが長期にわたって 1 を示している場合は、クラスターを評価してスケールアップまたはスケールアウトする必要があるかどうかを判断します。

詳細については、「[メトリクス](#)」ページの TrafficManagementActive メトリクスを参照してください。

メトリクスの統計と期間の選択

CloudWatch では、各メトリクスの統計および期間を選択できますが、すべての組み合わせが役に立つとは言えません。たとえば、CPUUtilization の Average、Minimum、および Maximum 統計は役に立ちますが、Sum 統計は役に立ちません。

ElastiCache のすべてのサンプルは、個々のキャッシュノードに対して 60 秒間発行されています。任意の 60 秒間において、キャッシュノードメトリクスに含まれるサンプルは 1 つだけです。

キャッシュノードのメトリクスを取得する方法の詳細については、「[CloudWatch クラスターとノードメトリクスのモニタリング](#)」を参照してください。

CloudWatch クラスターとノードメトリクスのモニタリング

ElastiCache と CloudWatch は、多様なメトリクスを収集できるように統合されています。CloudWatch を使用して、これらのメトリクスをモニタリングできます。

Note

次の例には、CloudWatch コマンドラインツールが必要です。CloudWatch の詳細と開発者ツールのダウンロードについては、「[CloudWatch 製品ページ](#)」を参照してください。

次の手順は、CloudWatch を使用して、過去 1 時間のキャッシュクラスターのストレージ領域統計を収集する方法を示しています。

Note

以下の例で指定されている StartTime 値と EndTime 値は、例示を目的としています。実際のキャッシュノードに適した開始時刻値および終了時刻値で置き換える必要があります。

ElastiCache 制限の詳細については、ElastiCache の「[AWS サービス制限](#)」を参照してください。

CloudWatch クラスターとノードメトリクスのモニタリング (コンソール)

キャッシュクラスターの CPU 使用率統計を収集するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。

2. メトリクスを表示するキャッシュノードを選択します。

Note

20 個を超えるノードを選択すると、コンソールでメトリクスを表示できなくなります。

- a. AWS マネジメントコンソールの [Cache Clusters] ページで、1 つ以上のキャッシュクラスターの名前をクリックします。

キャッシュクラスターの詳細ページが表示されます。

- b. ウィンドウ上部にある [Nodes] タブをクリックします。
- c. 詳細ウィンドウの [Nodes] タブで、メトリクスを表示するキャッシュノードを選択します。

使用可能な CloudWatch メトリクスのリストがコンソールウィンドウの下部に表示されます。

- d. [CPU Utilization] メトリクスをクリックします。

CloudWatch コンソールが開き、選択されたメトリクスが表示されます。[Statistic] および [Period] ドロップダウンリストボックスや [Time Range] タブを使用すると、表示されるメトリクスを変更できます。

CloudWatch CLI を使用した CloudWatch クラスターとノードメトリクスのモニタリング

キャッシュクラスターの CPU 使用率統計を収集するには

- Linux、macOS、Unix の場合:

```
aws cloudwatch get-metric-statistics \  
  --namespace AWS/ElastiCache \  
  --metric-name CPUUtilization \  
  --dimensions='[{"Name":"CacheClusterId","Value":"test"},  
{"Name":"CacheNodeId","Value":"0001"}]' \  
  --statistics=Average \  
  --start-time 2018-07-05T00:00:00 \  
  --end-time 2018-07-06T00:00:00 \  
  --period=3600
```

Windows の場合:

```
aws cloudwatch get-metric-statistics ^
  --namespace AWS/ElastiCache ^
  --metric-name CPUUtilization ^
  --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' ^
  --statistics=Average ^
  --start-time 2018-07-05T00:00:00 ^
  --end-time 2018-07-06T00:00:00 ^
  --period=3600
```

CloudWatch API を使用した CloudWatch クラスターとノードメトリックスのモニタリング

キャッシュクラスターの CPU 使用率統計を収集するには

- 以下のパラメータを指定して、CloudWatch API `GetMetricStatistics` を呼び出します (示されている開始時刻と終了時刻は例です。適切な開始時刻と終了時刻に置き換える必要があります)。
 - `Statistics.member.1=Average`
 - `Namespace=AWS/ElastiCache`
 - `StartTime=2013-07-05T00:00:00`
 - `EndTime=2013-07-06T00:00:00`
 - `Period=60`
 - `MeasureName=CPUUtilization`
 - `Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002`

Example

```
http://monitoring.amazonaws.com/
?Action=GetMetricStatistics
&SignatureVersion=4
&Version=2014-12-01
&StartTime=2018-07-05T00:00:00
&EndTime=2018-07-06T23:59:00
&Period=3600
&Statistics.member.1=Average
```

```
&Dimensions.member.1="CacheClusterId=mycachecluster"
&Dimensions.member.2="CacheNodeId=0002"
&Namespace=&AWS;/ElastiCache
&MeasureName=CPUUtilization
&Timestamp=2018-07-07T17%3A48%3A21.746Z
&AWS;AccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

Amazon SNS による ElastiCache イベントのモニタリング

重要なイベントがクラスター上で発生すると、ElastiCache から特定の Amazon SNS トピックに通知が送信されます。例には、ノードの追加の失敗、ノードの追加の成功、セキュリティグループの変更などが含まれます。主要イベントをモニタリングすることで、クラスターの現在の状態を知り、イベントに基づいて是正措置を取ることができます。

トピック

- [ElastiCache Amazon SNS 通知の管理](#)
- [ElastiCache イベントの表示](#)
- [イベント通知と Amazon SNS](#)

ElastiCache Amazon SNS 通知の管理

Amazon Simple Notification Service (Amazon SNS) を使用して重要なクラスターイベントの通知が送信されるように ElastiCache を設定できます。これらの例では、Amazon SNS トピックの Amazon リソースネーム (ARN) を使用してクラスターを設定し、通知を受け取るようにします。

Note

このトピックでは、Amazon SNS にサインアップし、Amazon SNS トピックをセットアップおよびサブスクライブしていることを前提としています。これを行う方法の詳細については、「[Amazon Simple Notification Service デベロッパーガイド](#)」を参照してください。

Amazon SNS トピックを追加する

以下のセクションでは、Amazon SNS トピックを AWS コンソール、AWS CLI、または ElastiCache API を使用して追加する方法について説明します。

Amazon SNS トピックを追加する (コンソール)

以下の手順は、クラスターの Amazon SNS トピックを追加する方法を示しています。レプリケーショングループの Amazon SNS トピックを追加するには、ステップ 2 で、クラスターを選択する代わりにレプリケーショングループを選択し、その後は同じステップに従います。

Note

このプロセスは、Amazon SNS トピックの変更に使用できます。

クラスターの Amazon SNS トピックを追加または変更するには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. [クラスター] で、Amazon SNS トピック ARN を追加または変更するクラスターを選択します。
3. [Modify] (変更) を選択します。
4. [Modify Cluster] の [Topic for SNS Notification] で、追加する SNS トピックを選択します。または、[Manual ARN input] を選択して Amazon SNS トピックの ARN を入力します。
5. [Modify] (変更) を選択します。

Amazon SNS トピックを追加する (AWS CLI)

クラスターの Amazon SNS トピックを追加または変更するには、AWS CLI コマンド `modify-cache-cluster` を使用します。

次のコード例では、Amazon SNS トピック ARN を [my-cluster] に追加します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-cluster \  
  --notification-topic-arn arn:aws:sns:us-  
west-2:123456789xxx:ElastiCacheNotifications
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^
```

```
--cache-cluster-id my-cluster ^  
--notification-topic-arn arn:aws:sns:us-west-2:123456789xx:ElastiCacheNotifications
```

詳細については、「[modify-cache-cluster](#)」を参照してください。

Amazon SNS トピックを追加する (ElastiCache API)

クラスターの Amazon SNS トピックを追加または変更するには、以下のパラメータを指定して `ModifyCacheCluster` アクションを呼び出します。

- `CacheClusterId=my-cluster`
- `TopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications`

Example

```
https://elasticache.amazonaws.com/  
?Action=ModifyCacheCluster  
&ApplyImmediately=false  
&CacheClusterId=my-cluster  
&NotificationTopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications  
&Version=2014-12-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20141201T220302Z  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Date=20141201T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

詳細については、「[ModifyCacheCluster](#)」を参照してください。

Amazon SNS 通知の有効化と無効化

クラスターでは、通知を有効または無効にすることができます。次の手順は、Amazon SNS 通知を無効にする方法を示しています。

Amazon SNS 通知の有効化と無効化 (コンソール)

AWS Management Console を使用して Amazon SNS 通知を無効にするには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. Redis を実行しているクラスターのリストを表示するには、左のナビゲーションペインで、[Redis] を選択します。
3. 通知を変更するクラスターの左側にあるボックスを選択します。
4. [Modify] (変更) を選択します。
5. In [Modify Cluster] の [Topic for SNS Notification], で、[Disable Notifications] を選択します。
6. [Modify] (変更) を選択します。

Amazon SNS 通知の有効化と無効化 (AWS CLI)

Amazon SNS 通知を無効にするには、以下のパラメータを指定して `modify-cache-cluster` コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \  
  --cache-cluster-id my-cluster \  
  --notification-topic-status inactive
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^  
  --cache-cluster-id my-cluster ^  
  --notification-topic-status inactive
```

Amazon SNS 通知の有効化と無効化 (ElastiCache API)

Amazon SNS 通知を無効にするには、以下のパラメータを指定して `ModifyCacheCluster` アクションを呼び出します。

- `CacheClusterId=my-cluster`
- `NotificationTopicStatus=inactive`

この呼び出しにより、以下のような出力が返されます。

Example

```
https://elasticache.us-west-2.amazonaws.com/  
  ?Action=ModifyCacheCluster  
  &ApplyImmediately=false  
  &CacheClusterId=my-cluster  
  &NotificationTopicStatus=inactive  
  &Version=2014-12-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
  &X-Amz-Date=20141201T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

ElastiCache イベントの表示

ElastiCache は、クラスターのインスタンス、セキュリティグループ、パラメータグループに関連するイベントを記録します。この情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などがあります。ElastiCache コンソール、AWS CLI `describe-events` コマンド、または ElastiCache API アクション `DescribeEvents` を使用して、ログから簡単にイベントを取得できます。

次の手順は、過去 24 時間 (1440 分) のすべての ElastiCache イベントを表示する方法を示しています。

ElastiCache イベントの表示 (コンソール)

次の手順は、ElastiCache コンソールを使用してイベントを表示します。

ElastiCache コンソールを使用してスタックイベントを表示するには

1. AWS Management Console にサインインして、ElastiCache コンソール (<https://console.aws.amazon.com/elasticache/>) を開きます。
2. 利用可能なすべてのイベントのリストを表示するには、ナビゲーションペインで [Events] を選択します。

[Events] 画面のリスト内の各行は 1 個のイベントを表し、イベントのソース、イベントの種類 (キャッシュクラスター、キャッシュパラメータグループ、キャッシュセキュリティグループ、キャッシュサブネットグループ)、イベントの GMT 時間、イベントの説明が表示されます。

[Filter] を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

ElastiCache イベントの表示 (AWS CLI)

AWS CLI を使用して ElastiCache イベントのリストを作成するには、`describe-events` コマンドを使用します。オプションパラメーターを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、最大 40 個のキャッシュクラスターイベントを一覧表示します。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

次のコードでは、過去 24 時間 (1440 分) のすべてのイベントを一覧表示します。


```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

describe-events のコマンドによる出力は次のようになります。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
{
  "Events": [
    {
      "SourceIdentifier": "my-mem-cluster",
      "SourceType": "cache-cluster",
      "Message": "Finished modifying number of nodes from 1 to 3",
      "Date": "2020-06-09T02:01:21.772Z"
    },
    {
      "SourceIdentifier": "my-mem-cluster",
      "SourceType": "cache-cluster",
      "Message": "Added cache node 0002 in availability zone us-west-2a",
      "Date": "2020-06-09T02:01:21.716Z"
    },
    {
      "SourceIdentifier": "my-mem-cluster",
      "SourceType": "cache-cluster",
      "Message": "Added cache node 0003 in availability zone us-west-2a",
      "Date": "2020-06-09T02:01:21.706Z"
    },
    {
      "SourceIdentifier": "my-mem-cluster",
      "SourceType": "cache-cluster",
      "Message": "Increasing number of requested nodes",
      "Date": "2020-06-09T01:58:34.178Z"
    },
    {
      "SourceIdentifier": "mycluster-0003-004",
      "SourceType": "cache-cluster",
      "Message": "Added cache node 0001 in availability zone us-west-2c",
      "Date": "2020-06-09T01:51:14.120Z"
    },
    {
      "SourceIdentifier": "mycluster-0003-004",
      "SourceType": "cache-cluster",
      "Message": "This cache cluster does not support persistence (ex:
      'appendonly'). Please use a different instance type to enable persistence.",
      "Date": "2020-06-09T01:51:14.095Z"
    }
  ]
}
```

```
  },
  {
    "SourceIdentifier": "mycluster-0003-004",
    "SourceType": "cache-cluster",
    "Message": "Cache cluster created",
    "Date": "2020-06-09T01:51:14.094Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-005",
    "SourceType": "cache-cluster",
    "Message": "Added cache node 0001 in availability zone us-west-2b",
    "Date": "2020-06-09T01:42:55.603Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-005",
    "SourceType": "cache-cluster",
    "Message": "This cache cluster does not support persistence (ex:
'appendonly'). Please use a different instance type to enable persistence.",
    "Date": "2020-06-09T01:42:55.576Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-005",
    "SourceType": "cache-cluster",
    "Message": "Cache cluster created",
    "Date": "2020-06-09T01:42:55.574Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-004",
    "SourceType": "cache-cluster",
    "Message": "Added cache node 0001 in availability zone us-west-2b",
    "Date": "2020-06-09T01:28:40.798Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-004",
    "SourceType": "cache-cluster",
    "Message": "This cache cluster does not support persistence (ex:
'appendonly'). Please use a different instance type to enable persistence.",
    "Date": "2020-06-09T01:28:40.775Z"
  },
  {
    "SourceIdentifier": "mycluster-0001-004",
    "SourceType": "cache-cluster",
    "Message": "Cache cluster created",
    "Date": "2020-06-09T01:28:40.773Z"
  }
}
```

```
    }  
  ]  
}
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「[describe-events](#)」を参照してください。

ElastiCache イベントの表示 (ElastiCache API)

ElastiCache イベントのリストを ElastiCache API を使用して生成するには、DescribeEvents アクションを使用します。オプションパラメーターを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードは、40 個の最新のキャッシュクラスターイベントを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeEvents  
&MaxRecords=40  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&SourceType=cache-cluster  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

次のコードは、過去 24 時間 (1440 分) のキャッシュクラスターイベントを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeEvents  
&Duration=1440  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&SourceType=cache-cluster  
&Timestamp=20150202T192317Z  
&Version=2015-02-02  
&X-Amz-Credential=<credential>
```

上記のアクションでは、次のような出力が生成されます。

```
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">  
  <DescribeEventsResult>
```

```
<Events>
  <Event>
    <Message>Cache cluster created</Message>
    <SourceType>cache-cluster</SourceType>
    <Date>2015-02-02T18:22:18.202Z</Date>
    <SourceIdentifier>mem01</SourceIdentifier>
  </Event>

(...output omitted...)

</Events>
</DescribeEventsResult>
<ResponseMetadata>
  <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
</ResponseMetadata>
</DescribeEventsResponse>
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「[DescribeEvents](#)」を参照してください。

イベント通知と Amazon SNS

ElastiCache では、キャッシュクラスターで重要なイベントが発生したときに、Amazon Simple Notification Service (SNS) を使用してメッセージを発行できます。この機能を使用すると、キャッシュクラスターの個々のキャッシュノードエンドポイントに接続されたクライアントコンピュータでサーバーリストを更新できます。

Note

価格の情報やAmazon SNS ドキュメントへのリンクを含む、Amazon Simple Notification Service (SNS) の詳細については、「[Amazon SNS 製品ページ](#)」を参照してください。

通知は、指定した Amazon SNS トピック に発行されます。通知の要件は以下のとおりです:

- ElastiCache 通知に対して設定できるトピックは 1 つだけです。
- Amazon SNS トピックを所有する AWS アカウントは、通知が有効になっているキャッシュクラスターを所有するアカウントと同じアカウントである必要があります。
- 発行先の Amazon SNS トピックは暗号化できません。

Note

暗号化された (保存された) Amazon SNS トピックをクラスターにアタッチできます。ただし、ElastiCache コンソールからのトピックのステータスは非アクティブと表示され、ElastiCache がメッセージをトピックにプッシュしたときに、クラスターとトピックの関連付けが効果的に解除されます。

- Amazon SNS トピックは、ElastiCache クラスターと同じリージョンに存在している必要があります。

ElastiCache イベント

以下の ElastiCache イベントにより、Amazon SNS 通知がトリガーされます。イベントの詳細については、「[ElastiCache イベントの表示](#)」を参照してください。

イベント名	メッセージ	説明
ElastiCache:AddCacheNodeComplete	ElastiCache:AddCacheNodeComplete : <i>cache-cluster</i>	キャッシュノードがキャッシュクラスターに追加され、使用可能になっています。
ElastiCache:AddCacheNodeFailed (使用できる IP アドレスが不足しているため)	ElastiCache:AddCacheNodeFailed : <i>cluster-name</i>	使用できる IP アドレスが不足しているため、キャッシュノードを追加できませんでした。
ElastiCache:CacheClusterParametersChanged	ElastiCache:CacheClusterParametersChanged : <i>cluster-name</i>	1 つ以上のキャッシュクラスターパラメータが変更されました。
ElastiCache:CacheClusterProvisioningComplete	ElastiCache:CacheClusterProvisioningComplete <i>cluster-name-0001-005</i>	キャッシュクラスターのプロビジョニングが完了し、キャッシュクラスター内のキャッシュノードが使用可能になりました。

イベント名	メッセージ	説明
ElastiCache:CacheClusterProvisioningFailed (ネットワーク状態に互換性がないため)	ElastiCache:CacheClusterProvisioningFailed : <i>cluster-name</i>	存在しない Virtual Private Cloud (VPC) に新しいキャッシュクラスターに起動する試みが行われました。
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : <i>cluster-name</i>	キャッシュクラスターのスケールアップが正常に完了しました。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : <i>cluster-name</i>	キャッシュクラスターのスケールアップが失敗しました。
ElastiCache:CacheClusterSecurityGroupModified	ElastiCache:CacheClusterSecurityGroupModified : <i>cluster-name</i>	以下のいずれかのイベントが発生しました。 <ul style="list-style-type: none"> • キャッシュクラスターに承認されたキャッシュセキュリティグループのリストが修正されました。 • 1つ以上の新しい EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループで承認されました。 • 1つ以上の EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループから取り消されました。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceStarted	ElastiCache:CacheNodeReplaceStarted : <i>cluster-name</i>	<p>ElastiCache が、キャッシュノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、キャッシュノードの置き換えを開始しました。</p> <div data-bbox="1068 590 1507 905"><p> Note</p><p>置き換えられたキャッシュノードの DNS エントリは変更されません。</p></div> <p>ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のキャッシュクライアントライブラリは、ElastiCache がキャッシュノードを置き換えた後もキャッシュノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。</p>

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceComplete	ElastiCache:CacheNodeReplaceComplete : <i>cluster-name</i>	<p>ElastiCache が、キャッシュノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、キャッシュノードの置き換えを完了しました。</p> <div data-bbox="1068 590 1507 905" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"><p> Note</p><p>置き換えられたキャッシュノードの DNS エントリは変更されません。</p></div> <p>ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のキャッシュクライアントライブラリは、ElastiCache がキャッシュノードを置き換えた後もキャッシュノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。</p>

イベント名	メッセージ	説明
ElastiCache:CacheNodesRebooted	ElastiCache:CacheNodesRebooted : <i>cluster-name</i>	1つ以上のキャッシュノードが再起動されました。 メッセージ (Memcached): "Cache node %s shutdown" 2番目のメッセージ: "Cache node %s restarted"
ElastiCache:CertificateRenewalComplete (Redis のみ)	ElastiCache:CertificateRenewalComplete	Amazon CA 証明書が正常に更新されました。
ElastiCache:CreateReplicationGroupComplete	ElastiCache:CreateReplicationGroupComplete : <i>cluster-name</i>	レプリケーショングループが正常に作成されました。
ElastiCache>DeleteCacheClusterComplete	ElastiCache>DeleteCacheClusterComplete : <i>cluster-name</i>	キャッシュクラスターと関連するすべてのアプリケーションキャッシュノードの削除が完了しました。
ElastiCache:FailoverComplete (Redis のみ)	ElastiCache:FailoverComplete : <i>mycluster</i>	レプリカノードへのフェイルオーバーが成功しました。
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : <i>cluster-name-0001-005</i>	クラスタ内のレプリカの数が増加しました。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : <i>cluster-name-0003-004</i>	クラスターにレプリカを追加するプロセスが開始されました。
ElastiCache:NodeReplacementCanceled	ElastiCache:NodeReplacementCanceled : <i>cluster-name</i>	置き換え対象となっていたクラスター内のノードが置き換え対象ではなくなりました。
ElastiCache:NodeReplacementRescheduled	ElastiCache:NodeReplacementRescheduled : <i>cluster-name</i>	以前置き換え対象になったクラスター内のノードのスケジュールが、通知に記載されている新しい期間に変更されました。 実行可能なアクションについては、「 ノードの置換 」を参照してください。
ElastiCache:NodeReplacementScheduled	ElastiCache:NodeReplacementScheduled : <i>cluster-name</i>	クラスター内のノードが、通知に記載されている期間中の置き換え対象となりました。 実行可能なアクションについては、「 ノードの置換 」を参照してください。
ElastiCache:RemoveCacheNodeComplete	ElastiCache:RemoveCacheNodeComplete : <i>cluster-name</i>	キャッシュノードがキャッシュクラスターから削除されました。
ElastiCache:ReplicationGroupScalingComplete	ElastiCache:ReplicationGroupScalingComplete : <i>cluster-name</i>	レプリケーショングループのスケールアップオペレーションが正常に完了しました。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupScalingFailed	"Failed applying modification to cache node type to %s."	レプリケーショングループのスケールアップが失敗しました。
ElastiCache:ServiceUpdateAvailableForNode	"Service update is available for cache node %s."	セルフサービス更新は、ノードで使用できます。
ElastiCache:SnapshotComplete (Redis のみ)	ElastiCache:SnapshotComplete : <i>cluster-name</i>	キャッシュスナップショットの作成が正常に完了しました。
ElastiCache:SnapshotFailed (Redis のみ)	SnapshotFailed : <i>cluster-name</i>	<p>キャッシュスナップショットの作成に失敗しました。詳細な原因については、クラスターのキャッシュイベントを参照してください。</p> <p>スナップショットを表示する場合は、「DescribeSnapshots」を参照してください。ステータスは failed です。</p>

関連トピック

- [ElastiCache イベントの表示](#)

AWS CloudTrail を使用した Amazon ElastiCache API コール

Amazon ElastiCache は、AWS CloudTrail と統合されています。これは、Amazon ElastiCache のユーザー、ロール、または AWS のサービスで実行されたアクションを記録するためのサービスです。CloudTrail は、Amazon ElastiCache コンソールからの呼び出しと Amazon ElastiCache API オペレーションへのコード呼び出しを含む、Amazon ElastiCache の API コールをイベントとしてキャプチャします。証跡を作成する場合は、Amazon ElastiCache 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 ElastiCache に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、[AWS CloudTrail ユーザーガイド](#)を参照してください。

CloudTrail の Amazon ElastiCache 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。Amazon ElastiCache でアクティビティが発生すると、そのアクティビティは、[Event history] (イベント履歴) にある他の AWS のサービスのイベントとともに、CloudTrail イベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「[Viewing Events with CloudTrail Event History](#)」(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 ElastiCache のイベントなど、AWS アカウントのイベントを継続的に記録するには、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は AWS パーティションのすべてのリージョンからのイベントをログに記録し、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 ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- [追跡を作成するための概要](#)
- [CloudTrail のサポート対象サービスと統合](#)
- [Amazon SNS の CloudTrail の通知の設定](#)
- 「[複数のリージョンから CloudTrail ログファイルを受け取る](#)」および「[複数のアカウントから CloudTrail ログファイルを受け取る](#)」

すべての Amazon ElastiCache アクションは CloudTrail が記録します。これらのアクションは、[ElastiCache API リファレンス](#)で説明されています。例えば、CreateCacheCluster、DescribeCacheCluster、ModifyCacheClusterの各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「[CloudTrail userIdentity エlement](#)」を参照してください。

Amazon ElastiCache ログファイルエントリの理解

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、CreateCacheCluster アクションを示す CloudTrail ログエントリの例です。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EXAMPLEEXAMPLEEXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/elasticache-allow",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "elasticache-allow"
  },
  "eventTime": "2014-12-01T22:00:35Z",
  "eventSource": "elasticache.amazonaws.com",
  "eventName": "CreateCacheCluster",
  "awsRegion": "us-west-2",
```

```
"sourceIPAddress":"192.0.2.01",
"userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
"requestParameters":{
  "numCacheNodes":2,
  "cacheClusterId":"test-memcached",
  "engine":"memcached",
  "aZMode":"cross-az",
  "cacheNodeType":"cache.m1.small",
},
"responseElements":{
  "engine":"memcached",
  "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
  "cacheParameterGroup":{
    "cacheParameterGroupName":"default.memcached1.4",
    "cacheNodeIdsToReboot":{
    },
    "parameterApplyStatus":"in-sync"
  },
  "preferredAvailabilityZone":"Multiple",
  "numCacheNodes":2,
  "cacheNodeType":"cache.m1.small",

  "cacheClusterStatus":"creating",
  "autoMinorVersionUpgrade":true,
  "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
  "cacheClusterId":"test-memcached",
  "engineVersion":"1.4.14",
  "cacheSecurityGroups":[
    {
      "status":"active",
      "cacheSecurityGroupName":"default"
    }
  ],
  "pendingModifiedValues":{
  }
},
"requestID":"104f30b3-3548-11e4-b7b8-6d79ffe84edd",
"eventID":"92762127-7a68-42ce-8787-927d2174cde1"
}
```

次の例は、DescribeCacheCluster アクションを示す CloudTrail ログエントリです。Amazon ElastiCache のすべての Describe 呼び出し (Describe*) について、ResponseElements セクションが削除され、null と表示されます。

```
{
  "eventVersion":"1.01",
  "userIdentity":{
    "type":"IAMUser",
    "principalId":"EXAMPLEEXAMPLEEXAMPLE",
    "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
    "accountId":"123456789012",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "userName":"elasticache-allow"
  },
  "eventTime":"2014-12-01T22:01:00Z",
  "eventSource":"elasticache.amazonaws.com",
  "eventName":"DescribeCacheClusters",
  "awsRegion":"us-west-2",
  "sourceIPAddress":"192.0.2.01",
  "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
  "requestParameters":{
    "showCacheNodeInfo":false,
    "maxRecords":100
  },
  "responseElements":null,
  "requestID":"1f0b5031-3548-11e4-9376-c1d979ba565a",
  "eventID":"a58572a8-e81b-4100-8e00-1797ed19d172"
}
```

ModifyCacheCluster アクションを記録する CloudTrail のログエントリの例を以下に示します。

```
{
  "eventVersion":"1.01",
  "userIdentity":{
    "type":"IAMUser",
    "principalId":"EXAMPLEEXAMPLEEXAMPLE",
    "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
    "accountId":"123456789012",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "userName":"elasticache-allow"
  },
  "eventTime":"2014-12-01T22:32:21Z",
  "eventSource":"elasticache.amazonaws.com",
```

```
"eventName": "ModifyCacheCluster",
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.01",
"userAgent": "AWS CLI/ElastiCache 1.10 API 2014-12-01",
"requestParameters": {
  "applyImmediately": true,
  "numCacheNodes": 3,
  "cacheClusterId": "test-memcached"
},
"responseElements": {
  "engine": "memcached",
  "clientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
  "cacheParameterGroup": {
    "cacheParameterGroupName": "default.memcached1.4",
    "cacheNodeIdsToReboot": {
    },
    "parameterApplyStatus": "in-sync"
  },
  "cacheClusterCreateTime": "Dec 1, 2014 10:16:06 PM",
  "preferredAvailabilityZone": "Multiple",
  "numCacheNodes": 2,
  "cacheNodeType": "cache.m1.small",
  "cacheClusterStatus": "modifying",
  "autoMinorVersionUpgrade": true,
  "preferredMaintenanceWindow": "thu:05:00-thu:06:00",
  "cacheClusterId": "test-memcached",
  "engineVersion": "1.4.14",
  "cacheSecurityGroups": [
    {
      "status": "active",
      "cacheSecurityGroupName": "default"
    }
  ],
  "configurationEndpoint": {
    "address": "test-memcached.example.cfg.use1prod.cache.amazonaws.com",
    "port": 11211
  },
  "pendingModifiedValues": {
    "numCacheNodes": 3
  }
},
"requestID": "807f4bc3-354c-11e4-9376-c1d979ba565a",
"eventID": "e9163565-376f-4223-96e9-9f50528da645"
```



```
}
```

ElastiCache のクォータ

AWS アカウントには、AWS のサービスごとにデフォルトのクォータ (以前は制限と呼ばれていました) があります。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

ElastiCache のクォータを表示するには、[\[Service Quotas コンソール\]](#) を開きます。ナビゲーションペインで、[AWS services]、[ElastiCache] の順に選択します。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。Service Quotas でクォータがまだ利用できない場合は、[\[制限の引き上げ\]](#) のフォームを使用してください。

お客様の AWS アカウントには、ElastiCache に関連する以下のクォータがあります。

[リソース]	デフォルト値
リージョンごとのサーバーレスキャッシュ	40
キャッシュごとの 1 日あたりのサーバーレススナップショット	24
リージョンあたりのノード	300
インスタンスタイプごとの、クラスターあたりのノード (Redis クラスターモードが有効)	90
クラスターあたりのシャード (Redis クラスターモードが無効)	6
リージョンあたりのパラメータグループ	300
リージョンあたりのセキュリティグループ	50
リージョンあたりのサブネットグループ	300
サブネットグループあたりのサブネット	20
ユーザーグループあたりのユーザー	100

[リソース]	デフォルト値
ユーザーの最大数	1,000
ユーザーグループの最大数	100

リファレンス

このセクションのトピックでは、Amazon ElastiCache API および AWS CLI の ElastiCache セクションの使用について説明します。また、このセクションには一般的なエラーメッセージとサービス通知も含まれます。

- [ElastiCache API の使用](#)
- [ElastiCache API リファレンス](#)
- [AWS CLI リファレンスの ElastiCache セクション](#)
- [Amazon ElastiCache エラーメッセージ](#)
- [通知](#)

ElastiCache API の使用

このセクションでは、ElastiCache のオペレーションを使用および実装する方法を、メソッドに重点を置いて説明します。これらのオペレーションの詳細については、「[Amazon ElastiCache API リファレンス](#)」を参照してください。

トピック

- [クエリ API を使用する](#)
- [利用可能なライブラリ](#)
- [アプリケーションのトラブルシューティング](#)

クエリ API を使用する

クエリパラメータ

HTTP クエリベースのリクエストとは、HTTP 動詞 (GET または POST) とクエリパラメータ Action で記述する HTTP リクエストです。

各クエリリクエストに、アクションの認証と選択を処理するための一般的なパラメータがいくつか含まれている必要があります。

オペレーションの中にはパラメータのリストを取るものがあります。これらのリストは、`param.n` 表記を使用して指定されます。`n` 値は、1 から始まる整数です。

クエリリクエストの認証

HTTPS 経由でのみリクエストを送信できます。また、各クエリリクエストには署名を含める必要があります。このセクションでは、署名を作成する方法について説明します。次に説明する方法は、署名バージョン 4 と呼ばれます。

AWS へのリクエストを認証するために使用される基本的な手順を次に示します。この手順では、AWS に登録されており、アクセスキー ID とシークレットアクセスキーを持っていることを前提としています。

クエリ認証プロセス

1. 送信者は、AWS へのリクエストを構築します。
2. このトピックの次のセクションに示すように、送信者は、SHA-1 ハッシュ関数を使用してリクエストの署名 (Hash-based Message Authentication Code (HMAC) のキー付きハッシュ) を生成します。
3. リクエストの送信者は、リクエストデータ、署名、およびアクセスキー ID (使用するシークレットアクセスキーのキー識別子) を AWS に送信します。
4. AWS ではアクセスキー ID を使用して、シークレットアクセスキーを調べます。
5. AWS では、リクエストの署名を生成する際に使用したものと同一アルゴリズムを使い、リクエストデータとシークレットアクセスキーから署名を生成します。
6. 署名が一致すると、リクエストは認証されたものと見なされます。もし署名が一致しなかった場合、リクエストの処理は拒否され、AWS はエラーレスポンスを返します。

Note

リクエストに `Timestamp` パラメータが含まれている場合、リクエストに対して生成された署名はパラメータの値の 15 分後に期限が切れます。

リクエストに `Expires` パラメータが含まれている場合、署名は `Expires` パラメータで指定された時刻に期限が切れます。

リクエストの署名を計算するには

1. 本手順で後に必要となる、正規化されたクエリ文字列を作成します。

- a. 自然なバイト順のパラメータ名で、UTF-8 のクエリ文字列コンポーネントを並び替えます。パラメータは、GET URI または POST ボディから取得される場合があります。(Content-Type が application/x-www-form-urlencoded の場合)
 - b. URL は、以下の規則に応じてパラメータ名と値をエンコードします。
 - i. RFC 3986 が定義する非予約文字を、URL がエンコードすることはありません。非予約文字とは、A~Z、a~z、0~9、ハイフン (-)、アンダーバー (_)、ピリオド (.)、およびチルド (~) です。
 - ii. 他のすべての文字についても、%XY (X および Y には HEX 文字の 0-9 および大文字の A-F が入る) によるパーセントエンコードが必要です。
 - iii. パーセントは、拡張 UTF-8 文字を %XY%ZA... 形式でエンコードします。
 - iv. パーセントは、スペース文字を %20 (通常エンコードスキーマが行なうような + ではありません) としてエンコードします。
 - c. パラメータの値が空値の場合でも、エンコードされるパラメータ名とエンコードされる値の間に等号 (=) (ASCII コード 61) を入れます。
 - d. それぞれのパラメータ名と値のペアをアンド (&) (ASCII コード 38) で分割します。
2. 文字列を作成し、以下の擬似文法に従って ("\n" は ASCII 新規行を意味します) 署名を作成します。

```
StringToSign = HTTPVerb + "\n" +  
ValueOfHostHeaderInLowercase + "\n" +  
HTTPRequestURI + "\n" +  
CanonicalizedQueryString <from the preceding step>
```

HTTPRequestURI 要素は URI の HTTP 絶対パス要素ですが、クエリ文字列は含みません。HTTPRequestURI が空値の場合は、スラッシュ (/) を使用してください。

3. 作成したばかりの文字列を使い、シークレットアクセスキーをキーとして、また SHA256 または SHA1 をハッシュアルゴリズムとして、RFC 2104 に準拠した HMAC を計算します。

詳細については、<https://www.ietf.org/rfc/rfc2104.txt> を参照してください。

4. 結果の値を base64 に変換します。
5. その値は、Signature パラメータの値としてリクエストに含めます。

サンプルのリクエストを次に示します (見やすくするために改行が追加されています)。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterIdentifier=myCacheCluster  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-12-01
```

前のクエリ文字列では、次の文字列に対する HMAC 署名が生成されます。

```
GET\n  
elasticache.amazonaws.com\n  
Action=DescribeCacheClusters  
&CacheClusterIdentifier=myCacheCluster  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-12-01  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-west-2%2Felasticache  
%2Faws4_request  
&X-Amz-Date=20141201T223649Z  
&X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-  
amz-date  
content-type:  
host:elasticache.us-west-2.amazonaws.com  
user-agent:CacheServicesAPICommand_Client  
x-amz-content-sha256:  
x-amz-date:
```

結果の署名付きリクエストは次のようになります。

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&CacheClusterIdentifier=myCacheCluster  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-12-01  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-west-2/elasticache/aws4_request  
&X-Amz-Date=20141201T223649Z  
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date  
&X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

プロセスへの署名とリクエスト署名の生成の詳細については、トピック「[署名バージョン 4 の署名プロセス](#)」とそのサブトピックを参照してください。

利用可能なライブラリ

AWS では、クエリ API の代わりに言語固有の API を使用してアプリケーションを構築するソフトウェア開発者向け Software Development Kit (SDK) を提供します。こうした SDK には、リクエスト認証、リクエストの再実行、エラー処理など、(API には含まれない) 基本的な機能が用意されていて、簡単に開始できるようになっています。次のプログラミング言語の SDK と追加のリソースがあります。

- [Java](#)
- [Windows および .NET](#)
- [PHP](#)
- [Python](#)
- [Ruby](#)

他の言語については、「[サンプルコードとライブラリ](#)」を参照してください。

アプリケーションのトラブルシューティング

ElastiCache では、ElastiCache API とのやり取りで発生する問題をトラブルシューティングする際に役立つ、具体的でわかりやすいエラーを提供します。

エラーの取得

通常、アプリケーションでは、結果を処理する前にリクエストでエラーが生成されたかどうかを必ず確認します。エラーが発生したかどうかを確認する最も簡単な方法は、ElastiCache API からのレスポンスで Error ノードを検索することです。

XPath 構文を使用すると、簡単な方法で Error ノードがあるかどうかを検索し、エラーコードとメッセージを取得することができます。次のコードでは、Perl および XML::XPath モジュールによって、リクエスト時のエラーの発生を判定しています。エラーが発生した場合、レスポンス内の最初のエラーコードとメッセージが表示されます。

```
use XML::XPath;
```



```
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xml->findvalue("//Error[1]/Code"), "\n", " ",
$xml->findvalue("//Error[1]/Message"), "\n\n"; }
```

トラブルシューティングのヒント

ElastiCache API の問題を診断して解決するには、次の手順を実行することをお勧めします。

- ElastiCache が正しく実行されていることを確認します。

これを行うには、ブラウザウィンドウを開いて、ElastiCache サービス (<https://elasticache.amazonaws.com> など) に対してクエリリクエストを送信します。MissingAuthenticationTokenException または内部サーバーエラー 500 は、サービスが利用可能であり、リクエストに応答していることを示します。

- リクエストの構文を確認します。

「ElastiCache API リファレンス」には、各 ElastiCache オペレーションについてのリファレンスページがあります。パラメータを正しく使用していることをもう一度確認してください。間違っている可能性がある部分を判断するヒントとして、同様のオペレーションを実行しているサンプルのリクエストやユーザーシナリオを調べてください。

- フォーラムを確認します。

ElastiCache にはディスカッションフォーラムがあります。このフォーラムでは、これまで他のユーザーが経験してきた問題に対する解決策を探ることができます。フォーラムを見るには、以下をご覧ください。

<https://forums.aws.amazon.com/>

ElastiCache コマンドラインインターフェイスの設定

このセクションでは、コマンドラインツールを実行するための前提条件、コマンドラインツールを入手できる場所、ツールの設定方法と環境について説明します。また、このセクションにはツール使用の一般的な例も含まれています。

AWS CLI for ElastiCache を使用する場合にのみ、このトピックの手順に従います。

⚠ Important

Amazon ElastiCache コマンドラインインターフェイス (CLI) は、API バージョン 2014-09-30 以降の ElastiCache の機能強化をサポートしていません。コマンドラインから新しい ElastiCache 機能を使用するには、[AWS コマンドラインインターフェイス](#)を使用します。

トピック

- [前提条件](#)
- [コマンドラインツールを入手する](#)
- [ツールを設定する](#)
- [ツールでの認証情報の指定](#)
- [環境変数](#)

前提条件

この文書では、Linux/UNIX または Windows 環境での作業が可能であることを想定しています。Amazon ElastiCache コマンドラインツールは、UNIX ベースの環境である Mac OS X でも動作しますが、このガイドには Mac OS X に関する固有の手順は含まれていません。

慣例として、すべてのコマンドラインテキストには、一般的な **PROMPT>** コマンドラインプロンプトが前に付けられています。マシンの実際のコマンドラインプロンプトは、異なっている可能性があります。また、Linux/UNIX 固有のコマンドを示すには、**\$**、Windows 固有のコマンドを示すには、**C:\>** を使用します。コマンドからの出力例は、その直後にプレフィックスなしで表示されています。

Java ランタイム環境

このガイドで使用されているコマンドラインツールを実行するには、Java バージョン 5 以降が必要です。JRE または JDK のどちらでもかまいません。Linux/UNIX と Windows を含め各種プラットフォーム用の JRE は、[Java SE ダウンロード](#)から表示してダウンロードすることができます。

Java Home 変数の設定

コマンドラインツールは、Java ランタイムの場所を特定する環境変数 (`JAVA_HOME`) に応じて異なります。この環境変数は、`bin` 実行可能ファイル (Linux および UNIX の場合) または `java` 実行可

能ファイル (Windows の場合) が存在する `java.exe` という名前のサブディレクトリが含まれているディレクトリの絶対パスに設定する必要があります。

JAVA_HOME 変数の設定方法

1. Java ホーム変数を設定します。

- Linux と UNIX では、以下のコマンドを入力します。

```
$ export JAVA_HOME=<PATH>
```

- Windows は、以下のコマンドを入力します。

```
C:\> set JAVA_HOME=<PATH>
```

2. `$JAVA_HOME/bin/java -version` を実行して出力をチェックすることにより、パス設定を確認します。

- Linux/UNIX では、以下のような出力結果が表示されるはずですが、

```
$ $JAVA_HOME/bin/java -version
java version "1.6.0_23"
Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
```

- Windows では、以下のような出力結果が表示されるはずですが、

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.6.0_23"
Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
```

コマンドラインツールを入手する

コマンドラインツールは、「[ElastiCache 開発者ツールウェブサイト](#)」で ZIP ファイルとして入手できます。このツールは Java で作成されており、Windows 2000/XP/Vista/Windows 7 と Linux/UNIX、および Mac OSX 用のシェルスクリプトが含まれています。ZIP ファイルは自己完結型であ

り、インストール作業は不要です。ZIP ファイルをダウンロードして圧縮を解除し、ローカルマシン上の任意のディレクトリに保存してください。

ツールを設定する

コマンドラインツールは、サポートライブラリの場所の特定を環境変数 (AWS_ELASTICACHE_HOME) に依存しています。ツールを使うには、この環境変数を設定する必要があります。コマンドラインツールを解凍したディレクトリのパスに設定します。このディレクトリの名前は、ElastiCacheCli-A.B.nnnn (A、B、および n はバージョン/リリース番号) で、bin および lib という名前のサブディレクトリが含まれます。

AWS_ELASTICACHE_HOME 環境変数を設定するには

- コマンドラインウィンドウを開き、次のいずれかのコマンドを入力して、AWS_ELASTICACHE_HOME 環境変数を設定します。
 - Linux と UNIX では、以下のコマンドを入力します。

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

- Windows は、以下のコマンドを入力します。

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

ツールを使いやすくするため、ツールの BIN ディレクトリをシステム PATH に追加することをお勧めします。このガイドの残りの部分は、BIN ディレクトリがシステムパスに含まれていることを前提としています。

ツールの BIN ディレクトリをシステムパスに追加するには

- システム PATH にツールの BIN ディレクトリを追加するには、次のコマンドを入力します。
 - Linux と UNIX では、以下のコマンドを入力します。

```
$ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
```

- Windows は、以下のコマンドを入力します。

```
C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
```

Note

Windows の環境変数は、コマンドウィンドウを閉じたときにリセットされます。永続的に設定することもできます。詳細については、使用している Windows のバージョンのドキュメントを参照してください。

Note

スペースを含むパスは、二重引用符で囲む必要があります。たとえば、次のとおりです。
「C:\Program Files\Java」

ツールでの認証情報の指定

コマンドラインツールを使用するには、AWS アカウントで提供される AWS アクセスキーとシークレットアクセスキーが必要です。コマンドラインを使用するか、ローカルシステムにある認証情報ファイルから取得できます。

環境には、情報の編集に必要なテンプレートファイル `${AWS_ELASTICACHE_HOME}/credential-file-path.template` が含まれています。テンプレートファイルの内容は次のとおりです。

```
AWSAccessKeyId=<Write your AWS access ID>  
AWSSecretKey=<Write your AWS secret key>
```

Important

UNIX では、アクセス許可を認証情報ファイルの所有者に制限します。

```
$ chmod 600 <the file created above>
```

認証情報ファイルを設定したら、ElastiCache ツールで情報が検索されるように `AWS_CREDENTIAL_FILE` 環境変数を作成する必要があります。

AWS_CREDENTIAL_FILE 環境変数の設定方法

1. 環境変数を設定します:

- Linux/UNIX では、次のコマンドを使用して変数を更新します。

```
$ export &AWS;_CREDENTIAL_FILE=<the file created above>
```

- Windows では、次のコマンドを使用して変数を設定します。

```
C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
```

2. 設定が適切に機能することをチェックし、次のコマンドを実行します。

```
elasticache --help
```

すべての ElastiCache コマンドの使用方法ページが表示されます。

環境変数

環境変数はスクリプトやデフォルト値の設定またはそれらを一時的に上書きする場合に便利です。

AWS_CREDENTIAL_FILE 環境変数に加えて、ElastiCache コマンドラインインターフェイスに付属するほとんどの API ツールは次の変数をサポートしています。

- EC2_REGION — 使用する AWS リージョンです。
- AWS_ELASTICACHE_URL — サービス呼び出しに使用する URL です。EC2_REGION が指定されている場合または -- リージョンパラメータが渡されている場合は、別のリージョンのエンドポイントを指定する必要はありません。

以下の例は API ツールで使用するリージョンを設定する環境変数 EC2_REGION の設定方法を示します。

Linux、OS X、Unix

```
$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

Amazon ElastiCache エラーメッセージ

以下のエラーメッセージが Amazon ElastiCache によって返されます。ElastiCache、他の AWS のサービス、または Redis によって返される他のエラーメッセージを受け取る場合もあります。ElastiCache 以外のソースからのエラーメッセージの説明については、エラーメッセージ生成元のドキュメントを参照してください。

- [Cluster node quota exceeded](#)
- [Customer's node quota exceeded](#)
- [Manual snapshot quota exceeded](#)
- [Insufficient cache cluster capacity](#)

エラーメッセージ: Cluster node quota exceeded. Each cluster can have at most %n nodes in this region.

原因: クラスターで %n を超える数のノードが発生するようなクラスターの作成または変更を試みました。

解決策: リクエストを変更し、クラスターで %n を超える数のノードが発生しないようにします。または、%n を超える数のノードが必要な場合は、[Amazon ElastiCache ノードリクエストフォーム](#)を使用してリクエストを作成します。

詳細については、Amazon Web Services 全般のリファレンスの「[Amazon ElastiCache の制限](#)」を参照してください。

エラーメッセージ: Customer node quota exceeded. 最大 %n のノードをこのリージョンで持つことができます または、このリージョンの %s ノードのクォータに既に達しています。

原因: このリージョンのすべてのクラスター間で、アカウントに %n を超える数のノードが発生するようなクラスターの作成または変更を試みました。

解決策: リクエストを変更し、このアカウントのすべてのクラスター間のリージョンの合計ノード数が %n を超えないようにします。または、%n を超える数のノードが必要な場合は、[Amazon ElastiCache ノードリクエストフォーム](#)を使用してリクエストを作成します。

詳細については、Amazon Web Services 全般のリファレンスの「[Amazon ElastiCache の制限](#)」を参照してください。

エラーメッセージ: The maximum number of manual snapshots for this cluster taken within 24 hours has been reached または The maximum number of manual snapshots for this node taken within 24 hours has been reached its quota of %n

原因: 24 時間で許可される最大数の手動スナップショットをすでに作成している場合に、クラスターの手動スナップショットを作成しようとした。

解決策: 24 時間待ってから、クラスターの別の手動スナップショットを試みます。または、すぐに手動スナップショットを作成する必要がある場合は、クラスターの別のノードなど、同じデータがある別のノードのスナップショットを作成します。

エラーメッセージ: InsufficientCacheClusterCapacity

[原因]: 現在 AWS にはリクエストに対応するだけの十分なオンデマンド容量がありません。

解決策:

- 数分間待ってからリクエストを再度送信します。キャパシティーは頻繁に変化します。
- ノードやシャード (ノードグループ) の数を減らして新しいリクエストを送信します。たとえば、15 ノードを起動する 1 つのリクエストを行っている場合、代わりに 5 つのノードの 3 つのリクエストを作成するか、1 つのノードに対する 15 のリクエストを作成してみてください。
- クラスターを起動する場合は、アベイラビリティーゾーンを指定しないで新しいリクエストを送信します。
- クラスターを起動する場合は、別のノードタイプを使用して新しいリクエストを送信します (これは後でスケールアップできます)。詳細については、「[Redis ElastiCache のスケーリング](#)」を参照してください。

通知

このトピックでは、お客様が関心を持たれる可能性のある ElastiCache 通知に関して説明します。通知はほとんどの場合、一時的な状況またはイベントであり、ソリューションがみつかって、実装さ

れるまでの間持続します。通知には、開始日と解決の日付があり、その後は通知は関連付けられません。通知は、お客様に関連する場合も、しない場合もあります。実行するとクラスターのパフォーマンスを向上させる、実装のガイドラインをお勧めします。

通知は、新しいまたは改善された ElastiCache フィーチャーや機能は発表しません。

一般的な ElastiCache の通知

現在、エンジンに固有ではない未解決の ElastiCache の通知はありません。

ElastiCache for Redis 固有の通知

現在、未解決の ElastiCache for Redis の通知はありません。

ElastiCache for Redis ドキュメント履歴

- API バージョン: 2015-02-02
- ドキュメントの最新更新日: 2023 年 11 月 27 日

次の表は、2018 年 3 月以降の ElastiCache の各リリースにおける重要な変更点をまとめたものです。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

Redis 更新 ElastiCache の最近の

変更	説明	日付
ElastiCache for Redis が追加の C7gn ノードサイズのサポートを追加	ElastiCache for Redis で、追加の C7gn ノードサイズのサポートが追加されました。	2024 年 1 月 10 日
ElastiCache for Redis がサーバーレスキャッシュの作成をサポート	サーバーレスキャッシュを作成できるようになりました。これにより、キャッシュ管理が簡素化され、最も要求の厳しいアプリケーションにも対応できるよう即座にスケールできます。詳細については、「 デプロイオプションの選択 」を参照してください。この機能の一部として、サーバーレスキャッシュを管理対象の VPC エンドポイントに関連付けるための 新しいアクセス許可 が ElastiCacheServiceRolePolicy と AmazonElastiCacheFullAccess に追加されました。さらに、AmazonElastiCacheFullAccess ポリシーを使用するコンソー	2023 年 11 月 27 日

ルエクスペリエンスの改訂をサポートするためのアクセス許可が追加されました。

[ElastiCache for Redis がクラスターモードの変更をサポート](#)

クラスターモード無効 (CMD) からクラスターモード有効 (CME) にクラスターを移行できるようになりました。詳細については、「[クラスターモードの変更](#)」を参照してください。

2023 年 5 月 11 日

[ElastiCache for Redis が転送時の暗号化設定の変更をサポート](#)

クラスターを再構築または再プロビジョニングしたり、アプリケーションの可用性に影響を与えたりすることなく、Redis クラスターの TLS 設定を変更できるようになりました。詳細については、「[既存のクラスターで転送時の暗号化を有効にする](#)」を参照してください。

2022 年 12 月 28 日

[ElastiCache for Redis が IAM を使用したユーザーの認証をサポート](#)

IAM 認証では、AWS IAM ID を使用して ElastiCache for Redis への接続を認証できません。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。詳細については、「[IAM を使った認証](#)」を参照してください。

2022 年 11 月 16 日

[ElastiCache for Redis が Redis 7 をサポートするようになりました](#)

このリリースでは、Redis 関数、ACL の改善、シャードされた Pub/Sub など、いくつかの新機能が Amazon ElastiCache for Redis に追加されました。詳細については、「[for ElastiCache Redis バージョン 7.0](#)」を参照してください。

2022 年 11 月 8 日

[ElastiCache for Redis が IPv6 をサポートするようになりました](#)

ElastiCache はインターネットプロトコルバージョン 4 と 6 (IPv4 と IPv6) をサポートしているため、IPv4 接続のみ、IPv6 接続のみ、または IPv4 と IPv6 の両方の接続 (デュアルスタック) を受け入れるようにクラスターを設定できます。IPv6 は、[Nitro システム](#)上に構築されたすべてのインスタンスで Redis エンジンバージョン 6.2 以降を使用するワークロードでサポートされています。IPv6 ElastiCache 経由でにアクセスする場合、追加料金はかかりません。詳細については、「[ネットワークタイプの選択](#)」を参照してください。

2022 年 11 月 7 日

[ElastiCache for Redis がネイティブ JavaScript Object Notation \(JSON\) 形式をサポート](#)

ネイティブ JavaScript Object Notation (JSON) 形式は、Redis クラスター内の複雑なデータセットをエンコードするためのシンプルでスキーマレスな方法です。Redis クラスター内の JavaScript Object Notation (JSON) 形式を使用してデータをネイティブに保存してアクセスし、それらのクラスターに保存されている JSON データを更新できます。シリアル化および逆シリアル化するためのカスタムコードを管理する必要はありません。詳細については、「[JSON の使用開始](#)」を参照してください。

2022 年 5 月 25 日

[ElastiCache がサポートされるようになりました PrivateLink](#)

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、AWS Direct Connect 接続なしで API ElastiCache オペレーションにプライベートにアクセスできます。詳細については、「Redis 用の [Amazon ElastiCache API とインターフェイス VPC エンドポイント \(AWS PrivateLink\)](#)」または「Memcached 用の [Amazon ElastiCache API とインターフェイス VPC エンドポイント \(AWS PrivateLink\)](#)」を参照してください。

2022 年 1 月 24 日

[ElastiCache for Redis が Redis 6.2 とデータ階層化をサポート](#)

Amazon ElastiCache for Redis では、Amazon でサポートされている Redis エンジンの次のバージョンが導入されました ElastiCache。ElastiCache Redis 6.2 では、8 vCPUs 以上の x86 ノードタイプ、または 4 vCPU 以上の Graviton2 ノードタイプを使用する TLS vCPUs。ElastiCache for Redis では、データ階層化も導入されています。データ階層化は、クラスターを数百テラバイトの容量までスケールするための低コストな方法として使用できます。詳細については、[ElastiCache 「for Redis バージョン 6.2 \(拡張\)」](#) および [「データ階層化」](#) を参照してください。

2021 年 11 月 23 日

[Auto Scaling がサポートされました](#)

ElastiCache for Redis が Auto Scaling をサポートするようになりました。ElastiCache for Redis の Auto Scaling は、ElastiCache for Redis サービスに必要なシャードまたはレプリカを自動的に増減する機能です。ElastiCache は Application Auto Scaling サービスを活用してこの機能を提供します。詳細については、[「Auto Scaling ElastiCache for Redis クラスター」](#) を参照してください。

2021 年 8 月 19 日

[Redis スローログの配信がサポートされました](#)

ElastiCache では、Redis SLOWLOG を Amazon Data Firehose または Amazon CloudWatch Logs の 2 つの送信先のいずれかにストリーミングできるようになりました。詳細については、「[ログの配信](#)」を参照してください。

2021 年 4 月 22 日

[リソースおよび条件キーのタグ付けがサポートされました](#)

ElastiCache で、クラスターやその他の ElastiCache リソースの管理に役立つタグ付けがサポートされるようになりました。詳細については、「[ElastiCache リソースにタグを付ける](#)」を参照してください。では、条件キーのサポート ElastiCache も導入されています。IAM ポリシーを有効にする方法を決める条件を指定できます。詳細については、「[条件キーの使用](#)」を参照してください。

2021 年 4 月 7 日

[ElastiCache が AWS Outposts で利用可能になりました](#)

[AWS Outposts](#) は、ネイティブ AWS サービス、インフラストラクチャ、運用モデルをほぼすべてのデータセンター、コロケーションスペース、またはオンプレミス施設に導入します。Outposts ElastiCache に をデプロイして、クラウドと同様にオンプレミスでキャッシュを設定、運用、使用できます。詳細については、Redis 用の「[Outposts の使用](#)」または Memcached 用の「[Outposts の使用](#)」を参照してください。

2020 年 10 月 8 日

[ElastiCache が Redis 6 をサポートするようになりました](#)

Amazon ElastiCache for Redis では、Amazon でサポートされている Redis エンジンの次のバージョンが導入されています ElastiCache。このバージョンには、[ロールベースのアクセスコントロールを用いたユーザーの認証](#)、バージョンレスのサポート、クライアント側のキャッシュ、および大幅な運用の改善などが含まれます。詳細については、[ElastiCache 「for Redis バージョン 6.0 \(拡張\)」](#) を参照してください。

2020 年 10 月 7 日

[ElastiCache が Local Zones をサポートするようになりました](#)

ローカルゾーンは、AWS リージョンの拡張であり、地理的にユーザーに近い場所にあります。新しいサブネットを作成してローカルゾーンに割り当てることで、親 AWS リージョンからローカルゾーンに任意の Virtual Private Cloud (VPC) を拡張できます。詳細については、「[Local Zones の使用](#)」を参照してください。

2020 年 9 月 25 日

[ElastiCache for Redis が、Redis クラスター環境のスケーリングを最大 500 ノードまたは 500 シャードまでサポート](#)

Redis Cluster モードでは、複数のシャード間でデータを分割するために使用できる設定が可能になり、スケーラビリティ、パフォーマンス、および可用性が向上します。この機能は、すべての AWS リージョンで Amazon ElastiCache for Redis バージョン 5.0.6 以降、および Redis クラスター環境 ElastiCache では既存および新規のすべてので使用できます。詳細については、「[Redis のノードとシャード](#)」を参照してください。

2020 年 8 月 13 日

[ElastiCache がリソースレベルのアクセス許可をサポートするようになりました](#)

AWS Identity and Access Management (IAM) ポリシーで ElastiCache リソースを指定することで、ユーザーのアクセス許可の範囲を制限できるようになりました。詳細については、「[リソースレベルのアクセス許可](#)」を参照してください。

2020 年 8 月 12 日

[ElastiCache for Redis が追加の Amazon CloudWatch メトリクスを追加](#)

ElastiCache for Redis が、PubSubCmds および を含む新しい CloudWatch メトリクスをサポートするようになりましたHyperLogLogBasedCmds 。完全なリストについては、「[Redis のメトリクス](#)」を参照してください。

2020 年 6 月 10 日

[ElastiCache で ElastiCache クラスターの自動更新がサポートされるようになりました](#)

Amazon は、サービス更新の「推奨日による適用」が経過した後、ElastiCache クラスターの自動更新をサポートする ElastiCache ようになりました。ElastiCache は、メンテナンスウィンドウを使用して、該当するクラスターの自動更新をスケジュールします。詳細については、「[セルフサービスの更新](#)」を参照してください。

2020 年 5 月 13 日

[ElastiCache for Redis が Redis のグローバルデータストアをサポート](#)

Global Datastore for Redis 機能は、AWS リージョン間でフルマネージド型、高速、信頼性が高く、安全なレプリケーションを提供します。この機能を使用すると、ElastiCache for Redis のクロスリージョンリードレプリカクラスターを作成して、AWS リージョン全体で低レイテンシーの読み取りとディザスタリカバリを実現できます。グローバルデータストアを作成、変更、および記述できます。また、グローバルデータストアで AWS リージョンを追加または削除し、グローバルデータストア内で AWS リージョンをプライマリとして昇格させることもできます。詳細については、[「Global Datastore を使用した AWS リージョン間のレプリケーション」](#)を参照してください。

2020 年 3 月 16 日

[ElastiCache for Redis が Redis バージョン 5.0.6 をサポート](#)

詳細については、[ElastiCache for Redis バージョン 5.0.6 \(拡張\)](#)」を参照してください。

2019 年 12 月 18 日

[Amazon ElastiCache が T3-Standard キャッシュノードをサポート](#)

Amazon で次世代の汎用バースト T3-Standard キャッシュノードを起動できるようになりました ElastiCache。Amazon EC2 の T3-Standard インスタンスは、CPU パフォーマンスのベースラインレベルを提供し、発生したクレジットが使い果たされるまでいつでも CPU 使用率をバーストすることができます。詳細については、「[サポートされているノードの種類](#)」を参照してください。

2019 年 11 月 12 日

[Amazon は、既存の ElastiCache for Redis サーバーで AUTH トークンの変更をサポートする ElastiCache ようになりました](#)

ElastiCache for Redis 5.0.6 では、新しいトークンを設定およびローテーションすることで、認証トークンを変更できるようになりました。使用中のアクティブなトークンを変更できるようになりました。また、転送中の暗号化が有効、認証トークンなしで以前に設定された既存のクラスターに、最新のトークンを追加することもできます。これは、クライアントリクエストを中断せずにトークンを設定および更新できる 2 段階のプロセスです。この機能は現在、ではサポートされていません AWS CloudFormation。詳細については、「[Redis AUTH コマンドを使用したユーザーの認証](#)」を参照してください。

2019 年 10 月 30 日

[Amazon ElastiCache が Amazon EC2 上の Redis からのオンラインデータ移行をサポート](#)

オンライン移行を使用して、Amazon EC2 のセルフホスト Redis から Amazon にデータを移行できるようになりました ElastiCache。詳細については、「[へのオンライン移行 ElastiCache](#)」を参照してください。

2019 年 10 月 28 日

[ElastiCache for Redis](#) では、Redis クラスターモードのオンライン垂直スケーリングが導入されました。

シャードされた Redis クラスターをオンデマンドでスケールアップまたはスケールダウンできるようになりました。Redis ElastiCache の場合、クラスターは引き続きオンラインのまま、受信リクエストを処理します。詳細については、「[ノードタイプの変更によるオンライン垂直スケーリング](#)」を参照してください。

2019 年 8 月 20 日

[ElastiCache for Redis](#) で、ユーザーは Amazon ElastiCache for Redis クラスターに 1 つのリーダーエンドポイントを使用できるようになりました。

この機能により、すべての読み取りトラフィックを単一のクラスターレベルのエンドポイントを介して ElastiCache for Redis クラスターに転送し、負荷分散と可用性を高めることができます。詳細情報については、「[接続エンドポイントの検索](#)」を参照してください。

2019 年 6 月 13 日

[ElastiCache for Redis で、ユーザーが自分のスケジュールでサービスの更新を適用できるようにになりました](#)

この機能を使用すると、メンテナンス期間中ではなく、任意のタイミングで利用可能なサービスの更新が適用されるように選択できます。これにより、特にピーク時のビジネスフローにおけるサービスの中断が最小限に抑えられ、クラスターが [ElastiCache サポートされているコンプライアンスプログラムに参加している場合にコンプライアンスを維持できます](#)。詳細については、[「Amazon でのセルフサービスの更新 ElastiCache」](#) および [「Amazon のコンプライアンス検証 ElastiCache」](#) を参照してください。

2019 年 6 月 4 日

[ElastiCache スタンダード リザーブドインスタンス サービス: 一部前払い、全額前払い、前払いなし。](#)

リザーブドインスタンスでは、ElastiCache インスタンスタイプと AWS リージョンに基づいて、Amazon インスタンスを 1 年間または 3 年間柔軟に予約できます。詳細については、[「リザーブドノードによるコスト管理」](#) を参照してください。

2019 年 1 月 18 日

[ElastiCache for Redis が Redis クラスターあたり最大 250 ノードをサポート](#)

ノードまたはシャードの制限は、Redis クラスター ElastiCache のごとに最大 250 まで増やすことができます。詳細については、[「シャード」](#) を参照してください。

2018 年 11 月 19 日

[ElastiCache for Redis がすべての T2 ノードで自動フェイルオーバー、バックアップ、復元をサポート](#)

ElastiCache for Redis では、自動フェイルオーバー、スナップショットの作成、すべての T2 ノードでのバックアップと復元のサポートが導入されました。詳細については、「for [ElastiCache Redis Backup and Restore and Snapshot](#)」を参照してください。

2018 年 11 月 19 日

[ElastiCache for Redis での M5 および R5 ノードのサポート](#)

ElastiCache for Redis は、AWS Nitro System に基づく M5 および R5 ノード、汎用およびメモリ最適化インスタンスタイプをサポートするようになりました。詳細については、「[サポートされているノードの種類](#)」を参照してください。

2018 年 10 月 23 日

[動的に変化するリードレプリカ数のサポート](#)

ElastiCache for Redis では、クラスタのダウンタイムなしでクラスタからリードレプリカを追加および削除するサポートが追加されました。このリリースにおけるこれらの変更とその他の変更の詳細については、「for Redis ユーザーガイド」の「[レプリカ数の変更](#)」を参照してください。ElastiCache API リファレンス [IncreaseReplicaCount](#) の [DecreaseReplicaCount](#) 「」 および 「」 も参照してください。ElastiCache

2018 年 9 月 17 日

[FedRAMP への準拠の認定](#)

ElastiCache for Redis は、FedRAMP コンプライアンスの認定を受けています。詳細については、[「Amazon のコンプライアンス検証 ElastiCache」](#) を参照してください。

2018 年 8 月 30 日

[Redis \(クラスターモードが有効\) エンジンのアップグレード](#)

Amazon ElastiCache for Redis では、Redis (クラスターモードが有効) エンジンバージョンのアップグレードのサポートが追加されました。詳細については、[「エンジンバージョンのアップグレード」](#) を参照してください。

2018 年 8 月 20 日

[PCI DSS への準拠の認定](#)

ElastiCache for Redis は PCI DSS コンプライアンスの認定を受けています。詳細については、[「Amazon のコンプライアンス検証 ElastiCache」](#) を参照してください。

2018 年 7 月 5 日

[ElastiCache for Redis 4.0.10 のサポート](#)

ElastiCache for Redis では、暗号化とオンラインクラスターのサイズ変更の両方を 1 つのバージョンでサポートする Redis 4.0.10 がサポートされるようになりました。詳細については、[ElastiCache 「for Redis バージョン 4.0.10 \(拡張\)」](#) を参照してください。

2018 年 6 月 14 日

[ユーザーガイドの再編成](#)

1 つのElastiCache ユーザーガイドが再構成され、Redis ([ElastiCache Redis ユーザーガイド用](#)) と Memcached ([ElastiCache Memcached ユーザーガイド用](#)) のユーザーガイドが別々になりました。[AWS CLI コマンドリファレンス: elasticache](#) セクションのドキュメント構造と [Amazon ElastiCache API リファレンス](#) は変更されません。

2018 年 4 月 20 日

[EngineCPUUtilization メトリックスのサポート](#)

ElastiCache for Redis に、現在使用されている CPU の容量の割合を報告する新しいメトリクス EngineCPU Utilization が追加されました。詳細については、「[Redis のメトリクス](#)」を参照してください。

2018 年 4 月 9 日

次の表は、2018 年 3 月以前の Redis ユーザーガイドに対する重要な変更点をまとめたものです。ElastiCache

変更	説明	変更日
アジアパシフィック (大阪: ローカル) リージョンのサポート	ElastiCache は、アジアパシフィック (大阪ローカル) リージョンのサポートを追加しました。現在、アジアパシフィック (大阪) リージョンでは、1 つのアベイラビリティゾーンのみをサポートしていて、招待によってのみ利用できます。詳細については、次を参照してください。 <ul style="list-style-type: none"> サポートされるリージョン 	2018 年 2 月 12 日

変更	説明	変更日
	<ul style="list-style-type: none"> サポートされているキャッシュノードの種類 	
欧州 (パリ) のサポート	<p>ElastiCache は、欧州 (パリ) リージョンのサポートを追加しました。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> サポートされるリージョン サポートされているキャッシュノードの種類 	2017 年 12 月 18 日
中国 (寧夏) リージョンのサポート	<p>Amazon は、中国 (寧夏) リージョンのサポート ElastiCache を追加しました。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> サポートされるリージョン サポートされているキャッシュノードの種類 	2017 年 12 月 11 日
サービスにリンクされたロールのサポート	<p>このリリースの では、サービスにリンクされたロール (SLR) のサポート ElastiCache が追加されました。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> Amazon ElastiCache でのサービスにリンクされたロールの使用 アクセス許可を設定する (新規 ElastiCache ユーザーのみ) 	2017 年 12 月 7 日

変更	説明	変更日
R4 ノードタイプのサポート	<p>このリリース ElastiCache の追加では、でサポートされているすべての AWS リージョンで R4 ノードタイプがサポートされています ElastiCache。オンデマンドまたはリザーブドキャッシュノードとして R4 ノードタイプを購入できます。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • サポートされているキャッシュノードの種類 • Redis のノードタイプ固有のパラメータ 	2017 年 11 月 20 日
ElastiCache for Redis 3.2.10 とオンラインリシャードのサポート	<p>Amazon ElastiCache for Redis では ElastiCache、for Redis 3.2.10 のサポートが追加されました。ElastiCache for Redis では、オンラインクラスターのサイズ変更も導入されています。これにより、クラスターが受信 I/O リクエストを処理しながら、クラスターからシャードを追加または削除できます。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • オンラインクラスターのサイズ変更 • Redis 用のオンラインリシャードおよびシャードの再分散 (クラスターモードが有効) 	2017 年 11 月 9 日
HIPAA 適格性	<p>ElastiCache for Redis バージョン 3.2.6 は、クラスターで暗号化が有効になっている場合に、HIPAA 適格性が認定されるようになりました。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • Amazon のコンプライアンス検証 ElastiCache • Amazon ElastiCache のデータセキュリティ 	2017 年 11 月 2 日

変更	説明	変更日
ElastiCache for Redis 3.2.6 および暗号化のサポート	<p>ElastiCache では、2 つの暗号化機能を含む ElastiCache for Redis 3.2.6 のサポートが追加されました。</p> <ul style="list-style-type: none">• 転送時の暗号化では、データの転送時 (クラスターのノード間、クラスターとアプリケーション間など) にデータが必ず暗号化されます。• 保管時の暗号化では、同期やバックアップオペレーションの実行中にオンディスクデータが暗号化されます。 <p>詳細については、次を参照してください。</p> <ul style="list-style-type: none">• Amazon ElastiCache のデータセキュリティ• サポートされている ElastiCache for Redis のバージョン	2017 年 10 月 25 日
接続パターンのトピック	<p>ElastiCache ドキュメントでは、Amazon VPC の ElastiCache クラスターにアクセスするためのさまざまなパターンに関するトピックを追加しています。</p> <p>詳細については、ElastiCache ユーザーガイドの Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン を参照してください。</p>	2017 年 4 月 24 日

変更	説明	変更日
自動フェイルオーバーのテストのサポート	<p>ElastiCache では、レプリケーションをサポートする Redis クラスターでの自動フェイルオーバーのテストのサポートが追加されました。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • 「自動フェイルオーバーのテスト」 (ElastiCache ユーザーガイド) を参照してください。 • ElastiCache API リファレンスの TestFailover • AWS CLI リファレンスの「test-failover」 	2017 年 4 月 4 日
強化された Redis の復元	<p>ElastiCache は、クラスターのサイズ変更による Redis のバックアップと復元を強化しました。この機能は、バックアップの作成に使用されるクラスターよりも、シャード数が異なるクラスターへのバックアップの復元をサポートします。(API および CLI では、この機能は異なる数のシャードではなく異なる数のノードグループを復元できます)。この更新では、さまざまな Redis スロット設定もサポートされます。詳細については、「バックアップから新しいキャッシュへの復元」を参照してください。</p>	2017 年 3 月 15 日
新しい Redis メモリ管理パラメータ	<p>ElastiCache は、新しい Redis パラメータを追加します。これにより reserved-memory-percent、予約メモリの管理が容易になります。このパラメータは、ElastiCache for Redis のすべてのバージョンで使用できます。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • 予約メモリの管理 • Redis 3.2.4 の新しいパラメータ 	2017 年 3 月 15 日

変更	説明	変更日
欧州西部 (ロンドン) リージョンのサポート	<p>ElastiCache で、欧州 (ロンドン) リージョンのサポートが追加されました。現在、T2 および M4 のノードタイプのみサポートされています。詳細については、次を参照してください。</p> <ul style="list-style-type: none">• サポートされるリージョン• サポートされているキャッシュノードの種類	2016 年 12 月 13 日
カナダ (モントリオール) リージョンのサポート	<p>ElastiCache で、カナダ (モントリオール) リージョンのサポートが追加されました。現在、この AWS リージョンでは、ノードタイプ M4 と T2 のみがサポートされています。詳細については、次を参照してください。</p> <ul style="list-style-type: none">• サポートされるリージョン• サポートされているキャッシュノードの種類	2016 年 12 月 8 日
M4 および R3 のノードタイプのサポート	<p>ElastiCache では、南米 (サンパウロ) リージョンの R3 および M4 ノードタイプと、中国 (北京) リージョンの M4 ノードタイプのサポートが追加されました。詳細については、次を参照してください。</p> <ul style="list-style-type: none">• サポートされるリージョン• サポートされているキャッシュノードの種類	2016 年 11 月 1 日
米国東部 2 (オハイオ) リージョンのサポート	<p>ElastiCache は、M4, T22、および R3 ノードタイプの米国東部 (オハイオ) リージョン (米国東部 2) のサポートを追加します。R3 詳細については、次を参照してください。</p> <ul style="list-style-type: none">• サポートされるリージョン• サポートされているキャッシュノードの種類	2016 年 10 月 17 日

変更	説明	変更日
Redis クラスターのサポート	<p>ElastiCache では、Redis クラスター (拡張) のサポートが追加されました。Redis クラスターを使用しているお客様は、最大 15 のシャード (ノードグループ) 間にデータを分割できます。シャードごとに最大 5 のリードレプリカによるレプリケーションがサポートされています。Redis クラスターの自動フェイルオーバー回数は、以前のバージョンに比べて約 1/4 になっています。</p> <p>このリリースでは、マネジメントコンソールが再設計され、コンソールでは業界の使用状況に合った用語が使用されています。</p> <p>詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • Memcached と Redis の比較 • ElastiCache Redis のコンポーネントと機能用 — ノード、シャード、クラスター、レプリケーションのセクションでの注意 • ElastiCache for Redis の用語 	2016 年 10 月 12 日
M4 ノードタイプサポート	<p>ElastiCache では、でサポートされているほとんどの AWS リージョンで、ノードタイプの M4 ファミリーのサポートが追加されましたElastiCache。オンデマンドまたはリザーブドキャッシュノードとして M4 ノードタイプを購入できます。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • サポートされているキャッシュノードの種類 • Redis のノードタイプ固有のパラメータ 	2016 年 8 月 3 日

変更	説明	変更日
ムンバイリージョンのサポート	<p>ElastiCache で、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。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • サポートされているキャッシュノードの種類 • Redis のノードタイプ固有のパラメータ 	2016 年 6 月 27 日
スナップショットのエクスポート	<p>ElastiCache では、Redis スナップショットをエクスポートして、 の外部からアクセスできるようになりましたElastiCache。詳細については、次を参照してください。</p> <ul style="list-style-type: none"> • バックアップのエクスポート Amazon ElastiCache ユーザーガイドの • CopySnapshot 「Amazon ElastiCache API リファレンス」の「」 	2016 年 5 月 26 日
ノードタイプのスケールアップ	<p>ElastiCache では、Redis ノードタイプをスケールアップする機能が追加されました。詳細については、「Redis ElastiCache のスケーリング」を参照してください。</p>	2016 年 3 月 24 日
エンジンの簡単なアップグレード	<p>ElastiCache は、Redis キャッシュエンジンを簡単にアップグレードする機能を追加します。詳細については、「エンジンバージョンとアップグレード」を参照してください。</p>	2016 年 3 月 22 日
R3 ノードタイプのサポート	<p>ElastiCache では、中国 (北京) リージョンおよび南米 (サンパウロ) リージョンでの R3 ノードタイプのサポートが追加されました。詳細については、「サポートされているキャッシュノードの種類」を参照してください。</p>	2016 年 3 月 16 日

変更	説明	変更日
Lambda 関数 ElastiCache を使用した へのアクセス	Amazon VPC ElastiCache で にアクセスするための Lambda 関数の設定に関するチュートリアルを追加しました。詳細については、「 ElastiCache のチュートリアルと動画 」を参照してください。	2016 年 2 月 12 日
Redis 2.8.24 のサポート	ElastiCache では、Redis バージョン 2.8.24 のサポートが追加され、Redis 2.8.23 以降の改善が追加されました。バグ修正および不正なメモリアクセスのアドレスのログ記録を含む改善。詳細については、次を参照してください。 <ul style="list-style-type: none">• ElastiCache for Redis バージョン 2.8.24 (拡張)• Redis 2.8 リリースノート	2016 年 1 月 20 日
アジアパシフィック (ソウル) リージョンのサポート	ElastiCache では、t2、m3、および r3 ノードタイプのアジアパシフィック (ソウル) (ap-northeast-2) リージョンのサポートが追加されました。	2016 年 1 月 6 日
Amazon ElastiCache コンソールの変更。	新しいバージョンの Redis では、より良く安定したユーザーエクスペリエンスが提供されるため、Redis バージョン 2.6.13、2.8.6、および 2.8.19 は ElastiCache マネジメントコンソールに表示されなくなりました。他のオプションと詳細については、「 サポートされている ElastiCache for Redis のバージョン 」を参照してください。	2015 年 12 月 15 日

変更	説明	変更日
Redis 2.8.23 のサポート。	ElastiCache では、Redis バージョン 2.8.23 のサポートが追加され、Redis 2.8.22 以降に改善が追加されました。バグ修正と新しいパラメータ <code>close-on-slave-write</code> のサポートを含む改善。パラメータを有効にした場合、読み取り専用レプリカに書き込もうとするクライアントの接続は切断されます。詳細については、「 ElastiCache for Redis バージョン 2.8.23 (拡張) 」を参照してください。	2015 年 11 月 13 日

変更	説明	変更日
Redis 2.8.22 のサポート。	<p>ElastiCache では、Redis バージョン 2.8.22 のサポート ElastiCache が追加され、バージョン 2.8.21 以降の機能強化と改善が追加されました。改善には以下のものがあります。</p> <ul style="list-style-type: none">保存プロセスが実装され、利用可能なメモリが少なく分岐保存が失敗する場合に、保存が正常に行われるようになりました。追加の CloudWatch メトリクス — SaveInProgress および ReplicationBytes。部分同期を有効にするために、Redis パラメータ repl-backlog-size がすべてのクラスターに適用されるようになりました。 <p>変更および詳細についての詳細なリストについては、「ElastiCache for Redis バージョン 2.8.22 (拡張)」を参照してください。</p> <p>このドキュメントリリースには、ドキュメントの再編成と ElastiCache コマンドラインインターフェイス (CLI) ドキュメントの削除が含まれています。コマンドラインの使用については、「のAWS コマンドライン」を参照してください ElastiCache。</p>	2015 年 9 月 28 日
Redis 2.8.21 のサポート	<p>ElastiCache では、Redis バージョン 2.8.21 のサポートと、バージョン 2.8.19 以降の Redis の改善が追加されました。この Redis リリースでは、いくつかのバグ修正が行われています。詳細については、「Redis 2.8 リリースノート」を参照してください。</p>	2015 年 7 月 29 日

変更	説明	変更日
新しいトピック: 外部 ElastiCache からのアクセス AWS	の外部から ElastiCache リソースにアクセスする 方法に関する新しいトピックを追加しました AWS。詳細については、 「の外部 ElastiCache からのアクセス AWS」 を参照してください。	2015 年 7 月 9 日
ノード置き換え に関するメッ セージが追加さ れました	ElastiCache は、スケジュールされたノード交 換、NodeReplacementScheduled、ElastiCache NodeReplacementRescheduledおよび ElastiCac he: に関する 3 つのメッセージを追加しますElast iCacheNodeReplacementCanceled。 ノードの置き換えがスケジュールされている場 合に実行できる詳細とアクションについては、 ElastiCache「」の「」を参照してください イベン ト通知と Amazon SNS 。	2015 年 6 月 11 日

変更	説明	変更日
Redis v. 2.8.19 のサポート。	<p>ElastiCache では、Redis バージョン 2.8.19 のサポートと、バージョン 2.8.6 以降の Redis の改善が追加されました。このサポートによって以下が実現します。</p> <ul style="list-style-type: none"> Redis コマンド PFADD、PFCOUNT、および PFMERGE を含む HyperLogLog データ構造。 辞書式範囲のクエリと新しいコマンド ZRANGEBYLEX、ZLEXCOUNT、および ZREMRANGEBYLEX。 多くのバグ修正を導入しました。バックグラウンド保存 (bgsave) の子プロセスが予期せずに終了したときに、プライマリ SYNC の障害によりプライマリノードがレプリカノードに古いデータを送信することを防ぎます。 <p>の詳細については HyperLogLog、「Redis の新しいデータ構造: HyperLogLog」 を参照してください。</p> <p>PFADD、PFCOUNT、および PFMERGE の詳細については、Redis ドキュメント を参照して をクリックしますHyperLogLog。</p>	2015 年 3 月 11 日
コスト配分タグのサポート	ElastiCache では、コスト配分タグのサポートが追加されました。詳細については、 「コスト配分タグによるコストのモニタリング」 を参照してください。	2015 年 2 月 9 日
AWS GovCloud (米国西部) リージョンのサポート	ElastiCache で AWS GovCloud、(米国西部) (us-gov-west-1) リージョンのサポートが追加されました。	2015 年 1 月 29 日

変更	説明	変更日
欧州 (フランクフルト) リージョンのサポート	ElastiCache では、欧州 (フランクフルト) (eu-central-1) リージョンのサポートが追加されました。	2015 年 1 月 19 日
Redis レプリケーショングループのマルチ AZ サポート	ElastiCache は、プライマリノードから Redis レプリケーショングループのリードレプリカにマルチ AZ のサポートを追加します。はレプリケーショングループの正常性を ElastiCache モニタリングします。プライマリが失敗した場合、は自動的にレプリカをプライマリに ElastiCache 昇格し、はそのレプリカを置き換えます。詳細については、「 マルチ AZ による ElastiCache for Redis のダウンタイムの最小化 」を参照してください。	2014 年 10 月 24 日
AWS CloudTrail サポートされる API コールの ログ記録	ElastiCache では、を使用してすべての ElastiCache API コール AWS CloudTrail をログに記録するためのサポートが追加されました。詳細については、「 AWS CloudTrail を使用した Amazon ElastiCache API コール 」を参照してください。	2014 年 9 月 15 日
サポートされる新しいインスタンスサイズ	ElastiCache では、追加の汎用 (T2) インスタンスのサポートが追加されました。詳細については、「 パラメータグループを使用したエンジンパラメータの設定 」を参照してください。	2014 年 9 月 11 日
サポートされる新しいインスタンスサイズ	ElastiCache では、追加の汎用 (M3) インスタンスとメモリ最適化 (R3) インスタンスのサポートが追加されました。詳細については、「 パラメータグループを使用したエンジンパラメータの設定 」を参照してください。	2014 年 7 月 1 日

変更	説明	変更日
Redis クラスターのバックアップと復元	<p>このリリースでは、ElastiCache で Redis クラスターのスナップショットを作成し、これらのスナップショットを使用して新しいクラスターを作成できます。バックアップは特定の時点でのクラスターのコピーであり、クラスターメタデータと Redis キャッシュ内のすべてのデータで構成されます。バックアップは Amazon S3 に保存され、お客様はいつでもスナップショットから新しいクラスターにデータを復元できます。詳細については、「スナップショットおよび復元」を参照してください。</p>	2014 年 4 月 24 日
Redis 2.8.6	<p>ElastiCache は、Redis 2.6.13 に加えて Redis 2.8.6 もサポートしています。Redis 2.8.6 を使用すると、お客様は部分再同期のサポートによりリードレプリカの弾力性と耐障害性を高めることができ、常に使用可能にする必要があるリードレプリカのユーザー定義の最小数を増やすことができます。Redis 2.8.6 では、のフルサポートも提供されており publish-and-subscribe、サーバーで発生したイベントをクライアントに通知できます。</p>	2014 年 3 月 13 日

変更	説明	変更日
Redis キャッシュエンジン	<p>ElastiCache は、Memcached に加えて Redis キャッシュエンジンソフトウェアを提供しています。現在 Redis を使用しているお客様は、Redis ElastiCache スナップショットファイルから既存のデータを使用して新しい Redis キャッシュクラスタを「シード」できるため、マネージド ElastiCache 環境への移行が容易になります。</p> <p>Redis レプリケーション機能をサポートするために、ElastiCache API はレプリケーショングループをサポートするようになりました。お客様は、プライマリ Redis キャッシュノードを含むレプリケーショングループを作成し、プライマリノードのキャッシュデータと自動的に同期される1つ以上のリードレプリカノードを追加できます。読み込み量が多いアプリケーションは、リードレプリカにオフロードしてプライマリノードの負荷を軽減できます。リードレプリカは、プライマリキャッシュノード障害時のデータ損失から保護することもできます。</p>	2013 年 9 月 3 日
デフォルトの Amazon Virtual Private Cloud (VPC) サポート	<p>このリリースでは、ElastiCache は Amazon Virtual Private Cloud (VPC) と完全に統合されています。初めて使用する場合、キャッシュクラスタはデフォルトで Amazon VPC に作成されます。詳細については、「Amazon VPC と ElastiCache のセキュリティ」を参照してください。</p>	2013 年 1 月 8 日

変更	説明	変更日
Amazon Virtual Private Cloud (VPC) のサポート	このリリースでは、Amazon Virtual Private Cloud (VPC) で ElastiCache クラスターを起動できます。デフォルトでは、初めて使用する場合のキャッシュクラスターは Amazon VPC に自動的に作成されます。既存のお客様は、自分のペースで Amazon VPC に移行できます。詳細については、「 Amazon VPC と ElastiCache のセキュリティ 」を参照してください。	2012 年 12 月 20 日
新しいキャッシュノードタイプ	このリリースには、4 個の追加キャッシュノードタイプが用意されています。	2012 年 11 月 13 日
リザーブドキャッシュノード	このリリースは、リザーブドキャッシュノードのサポートを追加します。	2012 年 4 月 5 日
新規ガイド	これは、Amazon ElastiCache ユーザーガイドの最初のリリースです。	2011 年 8 月 22 日

AWS 用語集

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

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。