aws

ユーザーガイド

Amazon ElastiCache

API バージョン 2015-02-02

Amazon ElastiCache: ユーザーガイド

Table of Contents

Amazon とは ElastiCache

Amazon ElastiCache 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 ElastiCache は 2 つの形式で操作できます。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 ElastiCache は Valkey、RedisOSS、および Memcached エンジンで動作します。必要なエンジンがわからない場合は、このガイドの「Valkey、RedisOSS、および Memcached のセルフデザインキャッシュの比較」を参照してください。

サーバーレスキャッシュ

ElastiCache はサーバーレスキャッシュを提供するため、アプリケーションのキャッシュの追加と操作が簡素化されます。 ElastiCache サーバーレスを使用すると、可用性の高いキャッシュを 1 分以内に作成でき、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 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CLI。

ElastiCache 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 はキャッシュのメモリを常時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ネットワーク帯域幅、 と は、アプリケーションのニーズを満たすようにスケーリングされます。 は、デベロッパーにシンプルなエンドポイントエクスペリエンス ElastiCache 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で、ハードウェアプロビジョニング ElastiCache を管理します。 モニタリング、 ノードの置き換え、 ソフトウェアパッチを自動的かつ透過的に適用します。 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 Serverless は、Valkey 7.2、Redis OSS7.1 以降、および Memcached 1.6.21 以降と互換性があります。

独自の ElastiCache クラスターの設計

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 Valkey、Redis、OSSまたは Memcached クラスターを で設計 ElastiCache することを選択できます ElastiCache。これにより、クラスターの AWS アベイラビリティーゾーン間でノードタイプ、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、ソフトウェアパッチ適用を自動的に管理します。

独自の ElastiCache クラスターを設計することで、クラスターに対する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。Valkey、Redis、OSSまたは Memcached をクラスターモードで実行して水平スケーリングを有効にするか、クラスターモードなしで垂直スケーリングのみを実行することもできます。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。Valkey または Redis OSS 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を選択することもできます。

独自の ElastiCache クラスターを設計するときは、Valkey 7.2、Redis OSS4.0 ~ 7.1、または Memcached 1.4 以降を実行できます。

関連サービス

MemoryDB

ElastiCache または MemoryDB のどちらを使用するかを決定するときは、次の比較を考慮してください。

- ElastiCache は、Valkey、Redis、OSSまたは Memcached 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 (マイクロ秒 ElastiCache の読み取りおよび書き込みパフォーマンス) でデータアクセスを高速化するワークロードのキャッシュには、 を考慮する必要があります。また ElastiCache 、Valkey または Redis OSS データ構造を使用し、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APIsにアクセスするユースケースについても考慮する必要があります。
- ElastiCache 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削減するのに役立ちます。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、基盤となるデータベースをスケーリングするのではなく、 を使用することで、大規模で高速なパフォーマンスを実現し ElastiCache、データストレージコストを削減できます。

- MemoryDB は、超高速のプライマリデータベースを必要とするワークロード用の耐久性の高いメモリ内データベースです。Valkey および Redis と互換性がありますOSS。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。MemoryDB は、Valkey または Redis OSS データ構造とプライマリAPIsで耐久性のあるデータベースを使用してアプリケーションを構築する場合にも、ユースケースに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 Relational Database Service

ElastiCache 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削減できます。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、基盤となるデータベースをスケーリングするのではなく、 を使用することで、大規模で高速なパフォーマンスを実現し ElastiCache、データストレージコストを削減できます。

関連する Amazon Relational Database Service サービスの詳細については、「AmazonRDS」を参照してください。

ElastiCache 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削減できます。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、基盤となるデータベースをスケーリングするのではなく、 を使用することで、大規模で高速なパフォーマンスを実現し ElastiCache、データストレージコストを削減できます。

の ElastiCache 仕組み

ここでは、 ElastiCache デプロイの主要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メモリ内データストアです。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 には、サーバーレスクラスターと独自設計クラスターの 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」を参照してください。

> **Note**
>
> Amazon ElastiCache は Valkey、Redis、OSSおよび Memcached エンジンと連携します。必要なエンジンがわからない場合は、このガイドの「Valkey、RedisOSS、および Memcached のセルフデザインキャッシュの比較」を参照してください。

トピック

の ElastiCache 仕組み

ElastiCache サーバーレス

ElastiCache Serverless を使用すると、キャパシティプランニング、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、Valkey、Redis、OSSまたは Memcached クライアントで設定してキャッシュへのアクセスを開始できる単一のエンドポイントを受け取ります。

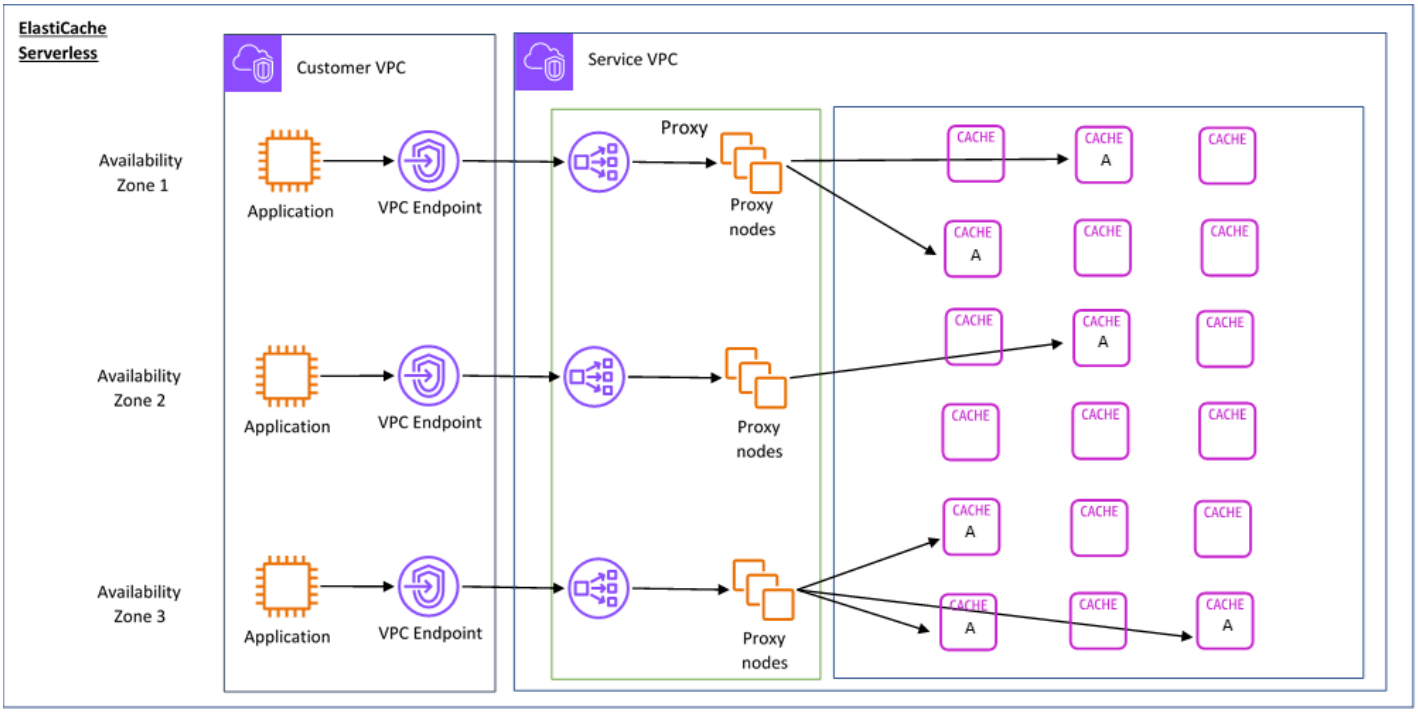
> **Note**
>
> - ElastiCache Serverless は、クラスターモードで Valkey、Redis、OSSまたは Memcached を実行し、 をサポートするクライアントとのみ互換性がありますTLS。

主な利点

- 容量計画なし： ElastiCache Serverless は、容量の計画を立て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にスケーリングします。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。
- P ay-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され、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 Serverless は、高可用性のために複数のアベイラビリティーゾーン (AZ) に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% SLAの可用性を提供します。
- 自動ソフトウェアアップグレード： ElastiCache Serverless は、アプリケーションに可用性に影響を与えることなく、キャッシュを最新のマイナーソフトウェア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 Serverless 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 で選択したサブネットに Virtual Private Cloud (VPC) エンドポイント ElastiCache を作成しますVPC。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ーのトポロジーとノードをクライアントから抽象化します。これにより、アプリケーションや接続をリセットすることなく、キャッシュノードを ElastiCache インテリジェントに負荷分散し、スケールアウトして追加し、キャッシュノードに障害が発生したときにキャッシュノードを置き換え、キャッシュノードのソフトウェアを更新できます。

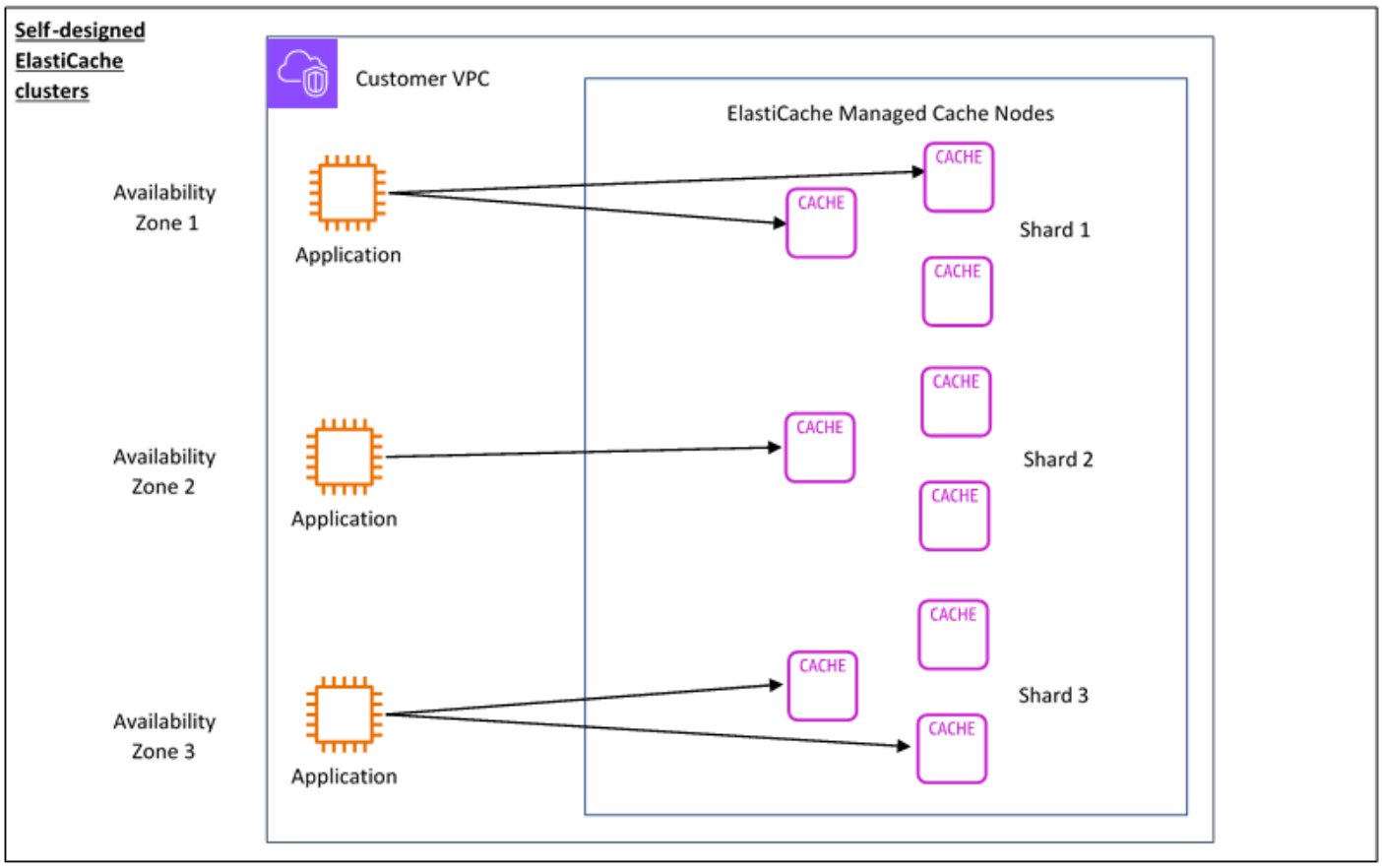
独自設計の ElastiCache クラスター

独自の ElastiCache クラスターを設計するには、クラスターのキャッシュノードファミリー、サイズ、ノード数を選択します。独自のクラスターを設計することで、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。Valkey または Redis OSSをクラスターモードで操作するには、複数のシャードを持つクラスターを作成するか、単一のシャードを持つクラスター以外のモードで操作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： では 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。または、複数のノードにまたがるクラスターを設計AZs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、 Valkey パラメータと Redis OSSパラメータまたは Memcached 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、キャッシュ上の CPUや Memory 使用量などのメトリクスに基づいてスケーリングを設定したりできます。

次の図は、 ElastiCache 独自設計のクラスターがどのように機能するかを示しています。



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 時間で保存されたデータの使用量と ElastiCache 処理単位 () で計算されるデータの使用量に対して料金が発生しますECPU。独自の ElastiCache クラスター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量の 1 時間あたりの料金が発生します。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

Serverless ElastiCache に保存されているデータは、ギガバイト時間 (GB 時間) で請求されます。ElastiCache サーバーレスは、キャッシュに保存されているデータを連続的にモニタリングし、1 分あたり複数回サンプリングし、時間平均を計算してキャッシュのデータストレージ使用量を 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、少なくとも 1 GB のデータが格納されているかどうか計測されます。

ElastiCache 処理単位 (ECPUs)

アプリケーションが ElastiCache 処理単位 (ECPUs) で ElastiCache サーバーレスで実行するリクエストに対して支払います。処理単位は、vCPU 時間と転送されたデータの両方を含む単位です。

- 単純な読み取りと書き込みには、転送されるデータのキロバイト (KB) ECPUごとに 1 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 を消費しますECPU。3.2 KB のデータを転送するSETリクエストは、3.2 を消費しますECPUs。
- Valkey と Redis ではOSS、2 つのディメンションのうち高い方ECPUsに基づいて、より多くの vCPU 時間を消費し、より多くのデータを転送するコマンドが消費されます。例えば、アプリケーションが HMGET コマンドを使用し、 が vCPU 時間の 3 倍を単純な SET/GET コマンドとして消費し、3.2 KB のデータを転送する場合、3.2 を消費しますECPU。または、2 KB のデータのみを転送した場合、3 を消費しますECPUs。
- Valkey と Redis ではOSS、追加の vCPU 時間を必要とするコマンドは、比例的により多くの を消費しますECPUs。例えば、アプリケーションが Valkey または Redis OSS HMGET コマンド を使用し、vCPU 時間の 3 倍を単純な SET/GET コマンドとして消費する場合、3 を消費しますECPUs。
- Memcached では、複数の項目で動作するコマンドは、比例的により多くの を消費しますECPUs。例えば、アプリケーションが 3 つの項目でマルチゲットを実行すると、3 を消費しますECPUs。
- Memcached では、より多くの項目で動作し、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高い方ECPUsに基づいて消費します。例えば、アプリケーションが GET コマンドを使用して 3 つの項目を取得し、3.2 KB のデータを転送する場合、3.2 を消費しますECPU。または、2 KB のデータのみを転送する場合、3 を消費しますECPUs。

ElastiCache サーバーレスは、ワークロードによってECPUs消費`ElastiCacheProcessingUnits`される を理解するのに役立つ という新しいメトリクスを出力します。

ノード時間

ノードファミリー、サイズ、EC2ノード数、アベイラビリティーゾーン間の配置を選択して、独自の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 バックアップ

バックアップは、 point-in-timeサーバーレスキャッシュのコピー、または Valkey または Redis が OSS独自に設計したクラスターです。ElastiCache では、いつでもデータのバックアップを取るか、

自動バックアップを設定できます。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 Valkey、Redis、OSSおよび Memcached 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リングします。 ElastiCache Serverless を使用すると、高可用性でスケー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、管理する必要はありません。 ElastiCache Serverless は、3 つのアベイラビリティーゾーンにデータを冗長的に自動的に保存し、99.99% の可用性のサービスレベルアグリーメント (SLA) を提供します。独自設計の Valkey または Redis OSSクラスターからのバックアップは、サーバーレス設定に復元できます。

独自設計型クラスター

Valkey、Redis OSS、または Memcached クラスターをきめ細かく制御する必要がある場合は、クラスターの AWS アベイラビリティーゾーン間でノードタイプ、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 で操作 ElastiCache ElastiCache できるように独自の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、ソフトウェアパッチ適用の管理に役立ちます。独自設計のクラスターは、最大 99.99% の可用性を提供するように設計されていますSLA。サーバーレス Valkey または Redis OSSキャッシュからのバックアップは、独自設計のクラスターに復元できます。

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。

- 新しいワークロードや予測が難しいワークロードのキャッシュを作成しています。
- 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。

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

- 既に ElastiCache Serverless を実行しており、Valkey、Redis、OSSまたは Memcached を実行しているノードのタイプ、ノードの数、およびそれらの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独自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	名前が 1 分未満の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。ユーザーはノードタイプ、ノード数、および AWS アベイラビリティーゾーン間の配置を選択できます
サポートされている ElastiCache バージョン	Valkey 7.2 以降、Redis OSSバージョン 7.1 以降、Memcached 1.6.21 以降	Valkey 7.2 以降、Redis OSSバージョン 4.0 以降、Memcached 1.4 以降
クラスターモード (バルキーおよび Redis OSS)	エンジンは `cluster mode enabled`でのみ動作します。クライアントは ElastiCache Serverless 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クラスターモードが無効で動作するように設定できます。

機能	サーバーレスキャッシュ	独自設計型クラスター
スケーリング	キャパシティ管理なしで、エンジンを垂直方向と水平方向に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、現在のキャパシティ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。 Valkey と Redis ではOSS、必要に応じてキャッシュノードのサイズを増減して垂直方向にスケーリングすることを選択できます。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することで、水平方向にスケールすることもできます。この機能は Memcached では使用できません。 Auto-Scaling 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やメモリ使用量などのメトリクスに基づいてスケーリングしたりできます。
クライアント接続	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、置き換え、アップグレード) をクライアントを切断することなく変更できるようになります。	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。

機能	サーバーレスキャッシュ	独自設計型クラスター
設定可能性	きめ細かな設定は利用できません。お客様は、キャッシュにアクセスできるサブネット、自動バックアップのオン/オフ、キャッシュの最大使用量制限など、基本的な設定を行うことができます。	独自設計のクラスターは、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、「エンジン固有のパラメータ」を参照してください。
マルチ AZ	データは複数のアベイラビリティーゾーンに非同期的に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。	単一のアベイラビリティーゾーンまたは複数のアベイラビリティーゾーン () にクラスターを設計するオプションを提供しますAZs。Valkey または Redis を使用する場合OSS、 はマルチ AZ クラスターに複数のアベイラビリティーゾーン間で非同期的にレプリケートされたデータを提供し、可用性を高め、読み取りレイテンシーを向上させます。
保管中の暗号化	常に有効になっています。お客様は、 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。	保管中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、お客様は 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。
転送中の暗号化 (TLS)	常に有効になっています。クライアントはTLS接続をサポートする必要があります。	を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	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。 Valkey バックアップと Redis OSSバックアップは互換性があり、 ElastiCache サーバーレスキャッシュまたは独自設計のクラスターに復元できます。	Valkey と Redis 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OSS。クラスターは、使用可能なリザーブドメモリによっては、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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。 Valkey バックアップと Redis OSSバックアップは互換性があり、 ElastiCache サーバーレスキャッシュまたは独自設計のクラスターに復元できます。

機能	サーバーレスキャッシュ	独自設計型クラスター
モニタリング	キャッシュヒット率、キャッシュミス率、データサイズ、ECPUs消費値などのキャッシュレベルメトリクスをサポートします。 ElastiCache Serverless は、キャッシュで重大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、およびアクションを選択できます 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。	ElastiCache 自己設計型クラスターは、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む、各ノードレベルでメトリクスを出力します。 独自設計のクラスターは、重要なイベントSNSの通知を送信します。「Memcached のメトリクス」および「Valkey と Redis のメトリクス OSS」を参照してください。
可用性	99.99% 可用性のサービスレベルアグリーメント (SLA)	独自設計のクラスターは、設定に応じて、最大 99.99% の可用性を実現するように設計されています (SLA)。

機能	サーバーレスキャッシュ	独自設計型クラスター
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。お客様はメジャーバージョンアップグレードの通知を受け取り、必要に応じて最新バージョンにアップグレードできます。	独自設計のクラスターは、マイナーバージョンやパッチ適用バージョンのアップグレード、メジャーバージョンのアップグレードに、カスタマー対応型のセルフサービスを提供します。マネージド更新は、ユーザー定義の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アップグレードまたはパッチバージョン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	Global Data Store をサポート。単一リージョンの書き込みと複数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、Valkey および Redis OSSワークロードに価格パフォーマンスオプションを提供します。
料金モデル	P ay-per-useは、GB 時間で保存されたデータと ElastiCache 処理単位 () のリクエストに基づいていますECPU。料金の詳細については、こちらを参照してください。	P ay-per-hourは、キャッシュノードの使用状況に基づきます。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

初めてのユーザー向けの Amazon ElastiCache リソース

初回のユーザーは、以下のセクションを読み、必要に応じて参照することをお勧めします。

- サービスのハイライトと料金 – 製品の詳細ページには、、 ElastiCacheサービスのハイライト、料金に関する一般的な製品概要が表示されます。
- ElastiCache 動画 – ElastiCache 動画このセクションには、Amazon を紹介する動画があります ElastiCache。このビデオでは、 の一般的なユースケース ElastiCache と、 を使用して

ElastiCacheアプリケーションのレイテンシーを短縮し、スループットを向上させる方法のデモについて説明します。

- [開始方法] – 「Amazon の開始方法 ElastiCache」セクションには、キャッシュクラスター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ー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ーを削除する方法も記載されています。
- 大規模なパフォーマンス – Amazon ホワイトペーパーを使用した大規模なパフォーマンス ElastiCacheは、アプリケーションの大規模なパフォーマンスを向上させるキャッシュ戦略に対処します。

前述のセクションを完了したら、これらのセクションを参照してください。

- ノードサイズの選択

  ノードは、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することができます。

- 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。

AWS Command Line Interface （ AWS CLI) を使用する場合は、以下のドキュメントを使用して開始できます。

- AWS Command Line Interface ドキュメント

  このセクションでは、 のダウンロード AWS CLI、システム上での AWS CLI の操作、 AWS 認証情報の提供について説明します。

- AWS CLI のドキュメント ElastiCache

  この個別のドキュメントでは、構文や例など、 ElastiCache コマンド AWS CLI のすべての について説明します。

アプリケーションプログラムを記述して、さまざまな一般的なプログラミング言語で ElastiCache APIを使用できます。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 は、 をサポートする多数のソフトウェア開発キット (SDKs) を提供しています ElastiCache。Java、.NET、、PHPRuby、およびその他の言語 ElastiCache を使用するためのコードを作成できます。これによりSDKs、リクエストを にフォーマットし ElastiCache、レスポンスを解析し、再試行ロジックとエラー処理を提供することで、アプリケーション開発を大幅に簡素化できます。

- の使用 ElastiCache API

  を使用しない場合は AWS SDKs、クエリ を使用して と ElastiCache 直接やり取りできますAPI。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

  この個別のドキュメントでは、構文や例を含むすべてのオペレーションについて説明します ElastiCache API。

AWS リージョンとアベイラビリティーゾーン

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 リージョンには、アベイラビリティーゾーンまたは と呼ばれる複数の個別の場所が含まれていますAZs。各アベイラビリティーゾーンは、他のアベイラビリティーゾーンの障害から分離されるように設計されています。各 は、同じ AWS リージョン内の他のアベイラビリティーゾーンに安価で低レイテンシーのネットワーク接続を提供するように設計されています。個別のアベイラビリティー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ーゾーンの選択」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ー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ー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 を使用した ElastiCache (Redis OSS) でのダウンタイムの最小化」を参照してください。

一般的な ElastiCache ユースケースと ElastiCache 役立つ方法

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でいいのでしょうか。最も最適化されたデータベースクエリやリモートAPIコールでも、メモリ内キャッシュからフラットキーを取得するよりも著しく遅くなります。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。

次の例は、 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 ElastiCache を向上させる方法の一部を示しています。

トピック

- インメモリデータストア
- ゲームリーダーボード
- メッセージング (Pub/Sub)
- レコメンデーションデータ (ハッシュ)
- ElastiCache 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つきます。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ァイルがあります。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もあります。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。

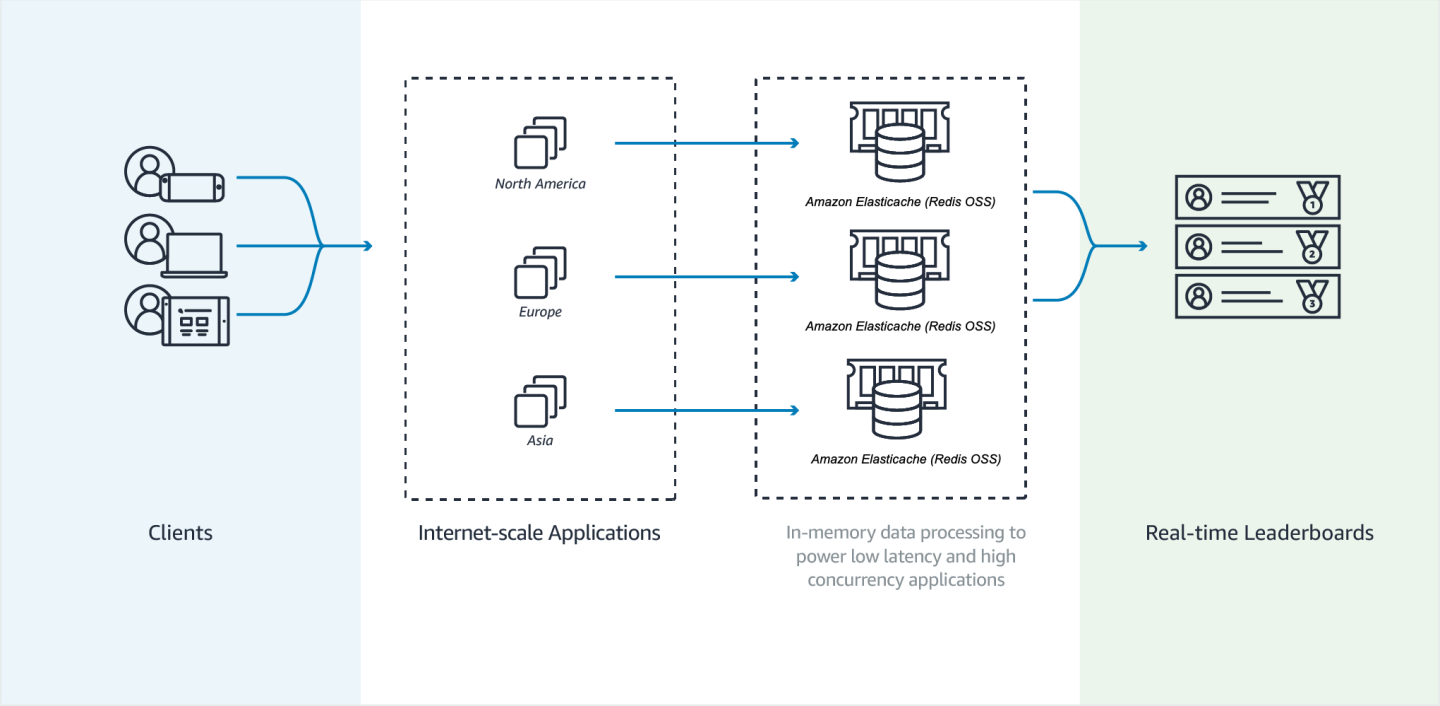
詳細については、「Memcached のキャッシュ戦略」を参照してください

ゲームリーダーボード

Valkey または Redis OSSソートセットを使用すると、リーダーボードの計算上の複雑さをアプリケーションからクラスターに移動できます。

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、大勢のプレー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。Valkey と Redis のOSSソートセットは、一意性と要素の順序付けの両方を保証します。ソートされたセットでは、新しい要素がソートされたセットに追加されるたびに、リアルタイムで再ランク付けされます。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。

次の図では、 ElastiCache ゲームリーダーボードの仕組みを示しています。



Example Valkey または Redis OSS 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、プレー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、プレー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 にすべてのプレーヤー間での自分のランクを知らせます。ランキングはゼロベースであるため、 は 2 番目のポジションにある 6 月の 1 ZREVRANKを返します。

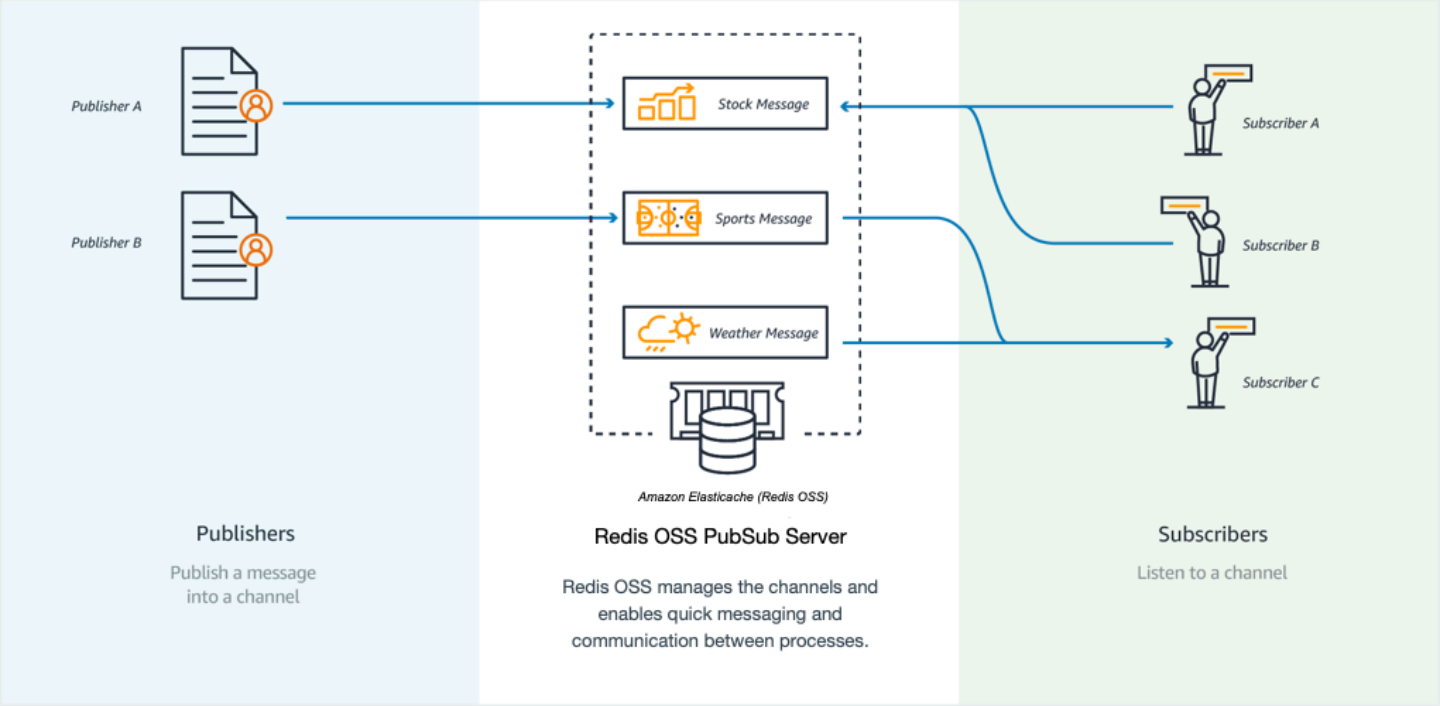
```
ZREVRANK leaderboard June
1
```

詳細については、ソートされたセットに関する Valkey ドキュメントを参照してください。

メッセージング (Pub/Sub)

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。Valkey および Redis OSSpub/sub パラダイムでは、誰が受信するかわからないメッセージを特定のチャネルに送信します。メッセージを受け取る人は、そのチャ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。

Pub/sub 機能は、キースペースとは関係ありません。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。次の図では、Valkey と Redis を使用したメッセージングの ElastiCache図を示していますOSS。



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、以前に使用したも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 つのチャンネルをサブスクライブするには、サブスクライブするチャンネルを指定する 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、サブスクライブを解除するチャネルを指定する UNSUBSCRIBE コマンドを使用して、チャネルへのサブスクリプション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

複数の特定のチャネルをサブスクライブするには、 SUBSCRIBE コマンドを使用してチャネルを一覧表示します。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サブスクリプションをキャンセルするには、 UNSUBSCRIBE コマンドを使用して、サブスクリプションを解除するチャンネルを指定します。

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

複数のチャネルへのサブスクリプションをキャンセルするには、 UNSUBSCRIBE コマンドを使用して、サブスクライブを解除するチャネルを指定します。

```
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ネルをサブスクライブできます。

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 コマンドと [P]UNSUBSCRIBE コマンドに送信されるチャネル文字列は一致する必要があります。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、サブスクライブされているチャネルに発行できません。

詳細については、Valkey ドキュメントの「Pub/Sub」を参照してください。

レコメンデーションデータ (ハッシュ)

Valkey または Redis DECRで INCR または を使用するとOSS、レコメンデーションのコンパイルが簡単になります。ユーザーが製品を「好き」になるたびに、item:productID:like に 1 を加えます。ユーザーが製品を「嫌い」になるたびに、item:productID:dislike に 1 を加えます。ハッシュを使用すると、製品を好きまたは嫌ったすべてのユーザーのリストを維持することもできます。

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
```

ElastiCache お客様の声

Airbnb、PBS、Esri などの企業が Amazon を使用してカスタマーエクスペリエンスを向上させてビジネスを ElastiCache 成長させる方法については、「How Others Use Amazon ElastiCache」を参照してください。

チュートリアルビデオで、追加のElastiCache カスタマーユースケースを確認することもできます。

Amazon の開始方法 ElastiCache

このセクションの実践的なチュートリアルを使用して、 の使用を開始して詳細を確認してください ElastiCache。

トピック

のセットアップ ElastiCache

ElastiCache ウェブサービスを使用するには、以下の手順に従います。

トピック

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/サインアップ 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 は、サインアッププロセスが完了した後に確認 E メールを送信します。https://aws.amazon.com/ に移動し、マイアカウント を選択すると、いつでも現在のアカウント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、 のセキュリティを確保し AWS アカウントのルートユーザー、 を有効にし AWS IAM Identity Center、管理ユーザーを作成して、日常的なタスクにルートユーザーを使用しないようにします。

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、「 ユーザーガイド」の AWS アカウント 「ルートユーザー (コンソール) の仮想MFAデバイスの有効化」を参照してください。 IAM

管理アクセスを持つユーザーを作成する

1. IAM Identity Center 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 Identity Center で、ユーザーに管理アクセスを許可します。

   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については、AWS IAM Identity Center ユーザーガイドの「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAMIdentity Center ユーザーの作成時に E メールアドレスにURL送信されたサインインを使用します。

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

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小権限のアクセス許可を適用するベストプラクティスに従う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。

プログラマチックアクセス権を付与する

ユーザーが の AWS 外部とやり取りする場合は、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
ワークフォースアイデンティティ （IAMアイデンティティセンターで管理されるユーザー）	一時的な認証情報を使用して AWS CLI、 AWS SDKs、または へのプログラムによるリクエストに署名します AWS APIs。	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。 • については AWS CLI、AWS Command Line Interface ユーザーガイドの「 を使用する AWS CLI ように を設定する AWS IAM Identity Center」を参照してください。 • AWS SDKs、ツール、および については AWS APIs、「 およびツールリファレンスガイド」のIAM「アイデンティティセンター認証」を参照してください。 AWS SDKs
IAM	一時的な認証情報を使用して AWS CLI、 AWS SDKs、または へのプログラムによるリクエストに署名します AWS APIs。	IAM 「 ユーザーガイド」のAWS 「リソースで一時的な認証情報を使用する」の手順に従います。
IAM	(非推奨) 長期認証情報を使用して、AWS CLI AWS SDKs、または へのプログラムによるリクエストに署名します AWS APIs。	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。 • 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。

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
		• および ツールについては AWS SDKs、AWS SDKs「 ツールリファレンスガイド」の「長期認証情報を使用した認証」を参照してください。 • については 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。 IAM

関連トピック:

- ユーザーガイド IAMの内容。 IAM
- 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。

- ID プロバイダーIAMを介して で管理されるユーザー :

  ID フェデレーションのロールを作成します。IAM ユーザーガイドの「サードパーティー ID プロバイダーのロールの作成 (フェデレーション)」の指示に従います。

- IAM ユーザー :
  - ユーザーが担当できるロールを作成します。「 ユーザーガイド」のIAM「ユーザーのロールを作成する」の手順に従います。 IAM

- (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」の手順に従います。

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があるか、サービスにリンクされたロールを作成iam:createServiceLinkedRole ElastiCacheしてい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)
- AWS Amazon の マネージドポリシー ElastiCache
- Amazon のサービスリンクロールの使用 ElastiCache

セットアップ EC2

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 インスタンスをまだお持ちでない場合は、「」でEC2インスタンスをセットアップする方法について説明します。 の使用開始。 EC2
- EC2 インスタンスはキャッシュと同じ にありVPC、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 はデフォルトにキャッシュ ElastiCache を作成しVPC、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に従うには、EC2インスタンスがデフォルトにありVPC、デフォルトのセキュリティグループがあることを確認します。

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アクセスを許可する

ElastiCache 自己設計型クラスターは Valkey コマンドと Redis OSS 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 Valkey または Redis OSS 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。

ElastiCache (Memcached) は 11211 および 11212 ポートを使用して Memcached コマンドを受け入れます。EC2 インスタンスから Memcached 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 のセキュリティグループを選択しますVPC。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 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. [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. Valkey または Redis を使用している場合はOSS、ポート範囲ボックスに と入力します6379。

    Memcached を使用している場合は、Port range ボックスに と入力します11211。

    e. ソースボックスで、ポート範囲 (0.0.0.0/0) がある任意の場所を選択して、Amazon 内で起動する Amazon EC2インスタンスをキャッシュVPCに接続できるようにします。

    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. タイプ 列で、カスタムTCPルールを選択します。

    h. ElastiCache (Redis OSS) を使用している場合は、Port range ボックスに と入力します6380。

ElastiCache (Memcached) を使用している場合は、Port range ボックスに と入力します11212。

i. ソースボックスで、ポート範囲 (0.0.0.0/0) がある任意の場所を選択して、Amazon 内で起動する Amazon EC2インスタンスをキャッシュVPCに接続できるようにします。

j. [保存] を選択します。

コマンドラインアクセスをダウンロードしてセットアップする

valkey-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。

Valkey ElastiCache で を使用する場合、valkey-cli ユーティリティが役立つ場合があります。Redis-cli で ElastiCache (Redis OSS) を使用している場合は、Redis OSSでも機能するため、valkey-cli に切り替えることを検討してください。

1. 任意の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て、valkey-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
   wget https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   tar xvzf valkey-7.2.6.tar.gz
   cd valkey-7.2.6
   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。

- valkey-cli または redis-cli をインストールするTLSときは、 のビルドサポートが必要です。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。

Valkey サーバーレスキャッシュを作成する

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

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. にサインイン AWS Management Console し、 を開きますhttps://console.aws.amazon.com/connect/。
2. コンソールの左側にあるナビゲーションペインで、Valkey キャッシュ を選択します。
3. コンソールの右側で、Valkey キャッシュの作成を選択します。
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ACTIVE「」ステータスになったら、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

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

Linux

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

Windows

```
aws elasticache create-serverless-cache ^
```

```
    --serverless-cache-name CacheName ^
    --engine valkey
```

[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完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、「キャッシュへのデータの読み取りと書き込み」に進みます。

キャッシュへのデータの読み取りと書き込み

このセクションでは、Amazon EC2インスタンスを作成し、接続できることを前提としています。これを行う方法については、「Amazon EC2 入門ガイド」を参照してください。

このセクションは、キャッシュに接続している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、およびEC2インスタンスの valkey-cli の設定があることも前提としています。このステップの詳細については、「のセットアップ ElastiCache」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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側にあるナビゲーションペインで、Valkey 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、 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 describe-serverless-caches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
```

Valkey キャッシュに接続する (Linux)

必要なエンドポイントができたので、EC2インスタンスにログインしてキャッシュに接続できます。次の例では、valkey-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/valkey-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"
```

Valkey キャッシュに接続する (Windows)

必要なエンドポイントができたので、EC2インスタンスにログインしてキャッシュに接続できます。次の例では、valkey-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き、Valkey または Redis OSS ディレクトリに変更してコマンドを実行します (注: Cache_Endpoint を前のステップで取得したエンドポイントに置き換えます) 。

```
c:\Valkey>valkey-cli -h Valkey_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、「(オプション) クリーンアップする」に進むことができます。

(オプション) クリーンアップする

作成した Amazon ElastiCache 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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側のナビゲーションペインで、Valkey Caches 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。

キャッシュが DELETINGステータスに移行するとすぐに、そのキャッシュに対して発生する料金が停止します。

AWS CLI

次の AWS CLI 例では、 コマンドを使用して delete-serverless-cache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
```

Status フィールドの値は に設定されますDELETING。

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

詳細については、以下のページ ElastiCache を参照してください。

- の使用 ElastiCache
- スケーリング ElastiCache
- Amazon でのログ記録とモニタリング ElastiCache
- 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
- ElastiCache イベントの Amazon SNSモニタリング

Valkey または Redis OSSサーバーレスキャッシュを作成する

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

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. にサインイン AWS Management Console し、 を開きますhttps://console.aws.amazon.com/connect/。
2. コンソールの左側にあるナビゲーションペインで、Valkey キャッシュまたは Redis OSSキャッシュ を選択します。
3. コンソールの右側で、Valkey キャッシュの作成または Redis OSSキャッシュの作成を選択します。
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ACTIVE「」ステータスになったら、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

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

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完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、「キャッシュへのデータの読み取りと書き込み」に進みます。

キャッシュへのデータの読み取りと書き込み

このセクションでは、Amazon EC2インスタンスを作成し、接続できることを前提としています。これを行う方法については、「Amazon EC2 入門ガイド」を参照してください。

このセクションは、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、およびEC2インスタンスの valkey-cli の設定があることも前提としています。このステップの詳細については、「のセットアップ ElastiCache」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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側にあるナビゲーションペインで、Valkey caches Redis OSS caches 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、 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 describe-serverless-caches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
```

Valkey または Redis OSS キャッシュに接続する (Linux)

必要なエンドポイントが得られたら、EC2インスタンスにログインしてキャッシュに接続できます。次の例では、valkey-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/valkey-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"
```

Valkey または Redis OSS キャッシュに接続する (Windows)

必要なエンドポイントが得られたら、EC2インスタンスにログインしてキャッシュに接続できます。次の例では、valkey-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き、Valkey ディレクトリに変更してコマンドを実行します (注: Cache_Endpoint を前のステップで取得したエンドポイントに置き換えます) 。

```
c:\Redis>valkey-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、「(オプション) クリーンアップする」に進むことができます。

(オプション) クリーンアップする

作成した Amazon ElastiCache 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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側のナビゲーションペインで、Valkey または Redis OSS Caches 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。

キャッシュが DELETINGステータスに移行するとすぐに、そのキャッシュに対して発生する料金が停止します。

AWS CLI

次の AWS CLI 例では、 コマンドを使用して delete-serverless-cache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
```

Status フィールドの値は に設定されますDELETING。

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

詳細については、以下のページ ElastiCache を参照してください。

- の使用 ElastiCache
- スケーリング ElastiCache
- Amazon でのログ記録とモニタリング ElastiCache
- 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
- ElastiCache イベントの Amazon SNSモニタリング

Memcached サーバーレスキャッシュを作成する

AWS Management Console

ElastiCache コンソールを使用して新しい Memcached サーバーレスキャッシュを作成するには：

1. にサインイン AWS Management Console し、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側のナビゲーションペインで、[Memcached キャッシュ] を選択します。
3. コンソールの右側で、[Memcached 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ACTIVE「」ステータスになったら、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。

を使用して新しいキャッシュを作成するには 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ます create-serverless-cache。

Linux

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

Windows

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

[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完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、「キャッシュへのデータの読み取りと書き込み」に進みます。

キャッシュへのデータの読み取りと書き込み

このセクションでは、Amazon EC2インスタンスを作成し、接続できることを前提としています。これを行う方法については、「Amazon EC2 入門ガイド」を参照してください。

デフォルトでは、 はデフォルトの にキャッシュ ElastiCache を作成しますVPC。EC2 インスタンスがキャッシュに接続できるようにVPC、インスタンスもデフォルトの に作成されていることを確認してください。

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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. コンソールの左側のナビゲーションペインで、[Memcached 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、 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 describe-serverless-caches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
```

Open を使用して接続するSSL

Open を使用して接続する方法についてはSSL、「」を参照してください。 ElastiCache 転送中の暗号化 (TLS)

Memcached Java クライアントを使用して接続する

Memcached Java クライアントを使用して接続する方法については、「ElastiCache 転送中の暗号化 (TLS)」を参照してください。

Memcached PHPクライアントを使用して接続する

```
<?php
$cluster_endpoint = "mycluster.serverless.use1.cache.amazonaws.com";
$server_port = 11211;

/* Initialize a persistent Memcached client in TLS mode */
$tls_client = new Memcached('persistent-id');
$tls_client->addServer($cluster_endpoint, $server_port);
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.serverless.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;
$tls_client->createAndSetTLSContext((array)$tls_config);

 /* store the data for 60 seconds in the cluster */
$tls_client->set('key', 'value', 60);
?>
```

Memcached Python クライアント (Pymemcache) を使用して接続する

https://pymemcache.readthedocs.io/en/latest/getting_started.html を参照してください。

```
import ssl
from pymemcache.client.base import Client

context = ssl.create_default_context()
cluster_endpoint = <To be taken from the AWS CLI / console>
target_port = 11211
```

```
memcached_client = Client(("{cluster_endpoint}", target_port), tls_context=context)
memcached_client.set("key", "value", expire=500, noreply=False)
assert self.memcached_client.get("key").decode() == "value"
```

Memcached NodeJS/TS クライアント (Electrode-IO memcache) を使用して接続する

https://github.com/electrode-io/memcache と https://www.npmjs.com/package/memcache-client を参照してください。

`npm i memcache-client` を用いたインストール

アプリケーションで、次のように memcached TLSクライアントを作成します。

```
var memcache = require("memcache-client");
const client = new memcache.MemcacheClient({server: "{cluster_endpoint}:11211", tls:
 {}});
client.set("key", "value");
```

Memcached Rust クライアント (rust-memcache) を使用して接続する

https://crates.io/crates/memcache と https://github.com/aisk/rust-memcache を参照してください。

```
// create connection with to memcached server node:
let client = memcache::connect("memcache+tls://<cluster_endpoint>:11211?
verify_mode=none").unwrap();

// set a string value
client.set("foo", "bar", 0).unwrap();
```

Memcached Go クライアント (Gomemcache) を使用して接続する

https://github.com/bradfitz/gomemcache を参照

```
c := New(net.JoinHostPort("{cluster_endpoint}", strconv.Itoa(port)))
c.DialContext = func(ctx context.Context, network, addr string) (net.Conn, error) {
var td tls.Dialer
td.Config = &tls.Config{}
return td.DialContext(ctx, network, addr)
}
foo := &Item{Key: "foo", Value: []byte("fooval"), Flags: 123}
err := c.Set(foo)
```

Memcached Ruby クライアント (Dalli) を使用して接続する

https://github.com/petergoldstein/「dalli」を参照してください。

```
require 'dalli'
ssl_context = OpenSSL::SSL::SSLContext.new
ssl_context.ssl_version = :SSLv23
ssl_context.verify_hostname = true
ssl_context.verify_mode = OpenSSL::SSL::VERIFY_PEER
client = Dalli::Client.new("<cluster_endpoint>:11211", :ssl_context => ssl_context);
client.get("abc")
```

Memcached を使用して接続します。NETクライアント (EnyimMemcachedCore)

https://github.com/cnblogs/EnyimMemcachedCore「��

```

"MemcachedClient": {
"Servers": [
{
"Address": "{cluster_endpoint}",
"Port": 11211
}
],
"UseSslStream":  true
}
```

これで、「(オプション) クリーンアップする」に進むことができます。

(オプション) クリーンアップする

の使用 AWS Management Console

次の手順では、デプロイから 1 つのキャッシュを削除します。複数のキャッシュを削除するには、削除するキャッシュごとに同じ手順を繰り返してください。別のキャッシュの削除手順を開始する前に、1 つのキャッシュの削除が終了するのを待つ必要はありません。

キャッシュを削除するには

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ElastiCache コンソールダッシュボードで、削除するキャッシュが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キャッシュが一覧表示されます。
3. 削除するキャッシュを選択するには、キャッシュのリストからキャッシュの名前を選択します。

   > **Important**
   >
   > ElastiCache コンソールから一度に削除できるキャッシュは 1 つだけです。複数のキャッシュを選択すると、削除オペレーションが無効になります。

4. [アクション] で、[削除] を選択します。
5. [キャッシュの削除] 確認画面で、[削除] を選択してキャッシュを削除するか、[キャンセル] を選択してクラスターを保持します。
6. [削除] を選択した場合は、キャッシュのステータスが [削除中] に変わります。

キャッシュが DELETINGステータスに移行するとすぐに、そのキャッシュに対する料金の発生が停止します。

の使用 AWS CLI

次のコードでは、キャッシュ my-cache を削除します。

```
aws elasticache delete-serverless-cache --serverless-cache-name my-cache
```

この delete-serverless-cacheCLIアクションは、1 つのサーバーレスキャッシュのみを削除します。複数のキャッシュを削除するには、削除するサーバーレスキャッシュごとに を呼び出し delete-

serverless-cacheます。1 つのサーバーレスキャッシュの削除が終了するまで待たなくても次のサーバーレスキャッシュを削除できます。

Linux、macOS、Unix の場合:

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

Windows の場合:

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

詳細については、 AWS CLI ElastiCache 「」トピックの「」を参照してください delete-serverless-cache。

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

詳細については、 ElastiCache 以下を参照してください。

- の使用 ElastiCache
- スケーリング ElastiCache
- のクォータ ElastiCache
- ElastiCache ベストプラクティスとキャッシュ戦略
- ElastiCache イベントの表示

チュートリアル: Python と の開始方法 ElastiCache

このセクションでは、Valkey と Redis ElastiCache の使用について学ぶのに役立つ実践的なチュートリアルについて説明しますOSS。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 SDKs 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。

トピック

- Python と ElastiCache

Python と ElastiCache

このチュートリアルでは、 for Python (Boto3) を使用して AWS SDK、次の ElastiCache (Redis OSS) 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 (Redis OSS) クラスターの作成 (クラスターモードが有効、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します。(この機能は、Valkey 7.2 以降、および Redis 6.0 OSS 以降で使用できます)。
- に接続 ElastiCache
- 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を進めるときは、 for Python (Boto) ドキュメントを参照してください AWS SDK。以下のセクションは、ElastiCache 低レベルクライアントに固有のものです ElastiCache。

チュートリアルの前提条件

- を使用するように AWS アクセスキーを設定します AWS SDKs。詳細については、「のセットアップ ElastiCache」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。

トピック

- チュートリアル: ElastiCache クラスターとユーザーの作成
- チュートリアル: への接続 ElastiCache
- 使用例

チュートリアル: ElastiCache クラスターとユーザーの作成

次の例では、boto3 を SDK ElastiCache (Redis OSS) 管理オペレーション (クラスターまたはユーザー作成) および redis-py/for data handling redis-py-cluster に使用します。

トピック

- クラスタモードが無効のクラスターを作成する
- TLS および を使用してクラスターモードが無効になっているクラスターを作成する RBAC
- クラスタモードが有効のクラスターの作成
- TLS および を使用してクラスターモードが有効になっているクラスターを作成する RBAC
- 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、CreateClusterModeDisabledCluster.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(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheCluste
 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/dg/
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 maximun 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='valkey',
        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, Valkey 7.2, one primary and two replicas
    elasticacheResponse = create_cluster_mode_disabled(
        #CacheNodeType='cache.m6g.large',
        EngineVersion='7.2',
        NumCacheClusters=3,
        ReplicationGroupDescription='Valkey cluster mode disabled with replicas',
        ReplicationGroupId='valkey202104053'
        )

    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、「でのクラスターの管理 ElastiCache」を参照してください。

TLS および を使用してクラスターモードが無効になっているクラスターを作成する RBAC

セキュリティを確保するために、Transport Layer Security (TLS) と Role-Based Access Control (RBAC) を使用して、クラスターモードが無効になっているクラスターを作成できます。トークンが

認証された場合OSSAUTH、すべての認証済み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 Valkey または Redis 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、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/dg/
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 maximun 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': 'valkey',
            '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, Valkey 7.2, 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='7.2',
        NumCacheClusters=3,
        ReplicationGroupDescription='Valkey cluster mode disabled with replicas',
        ReplicationGroupId='valkey202104',
        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、「でのクラスターの管理 ElastiCache」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、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/dg/
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 maximun 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='valkey',
        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.valkey7.2.cluster.on'
    )

    return response


# Creates a cluster mode enabled
response = create_cluster_mode_enabled(
    CacheNodeType='cache.m6g.large',
    EngineVersion='6.0',
    ReplicationGroupDescription='Valkey cluster mode enabled with replicas',
    ReplicationGroupId='valkey20210',
#   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、「でのクラスターの管理 ElastiCache」を参照してください。

TLS および を使用してクラスターモードが有効になっているクラスターを作成する RBAC

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になっているクラスターを作成するときに Transport Layer Security (TLS) と Role-Based Access Control (RBAC) を使用できます。トークンが認証された場合OSSAUTH、すべての認証済み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 Valkey または Redis 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、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.valkey7.2.clu
    """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/dg/
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 maximun 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': 'valkey',
            '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='7.2',
        ReplicationGroupDescription='Valkey cluster mode enabled with replicas',
        ReplicationGroupId='valkey2021',
    #   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、「でのクラスターの管理 ElastiCache」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

ではRBAC、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。特定のロール (管理者、人事) と連携したユーザーグループにユーザーを割り当て、1 つ以上の ElastiCache (Redis OSS) 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 Valkey または Redis OSSレプリケーショングループを使用してクライアント間のセキュリティ境界を確立し、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、UserAndUserGroups.py 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。

この例では、ユーザーのアクセス許可を持つアクセス文字列を使用します。アクセス文字列の詳細については、「」を参照してくださいアクセス文字列を使用したアクセス許可の指定。

```
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.
    :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
```

チュートリアル: への接続 ElastiCache

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

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

次のプログラムをコピーし、ConnectClusterModeDisabled.py 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。

```
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

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

```
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、 SDKの boto3 ElastiCache を使用して ElastiCache (Redis) を操作しますOSS。

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

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

```
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

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

```
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

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

```
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.pubsub()
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

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

```
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
 sucessfuly, acknowledge it and (optional) delete the message.
                    redis.xack(streamName,groupName,msgId)
                    logging.info("Stream message ID {} read and processed successfuly
 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: ElastiCache で にアクセスするように Lambda を設定する VPC

このチュートリアルでは、 ElastiCache サーバーレスキャッシュの作成、Lambda 関数の作成、Lambda 関数のテスト、オプションでクリーンアップを行う方法について説明します。

トピック

ステップ 1: ElastiCache 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。

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、 AWS Command Line Interface () を使用して、アカウントの us-east-1 リージョンVPCのデフォルトの Amazon にサーバーレスキャッシュを作成しますCLI。 ElastiCache コンソールまたは 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API、「」を参照してくださいValkey サーバーレス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 valkey
```

[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。キャッシュの作成 ElastiCache が完了するまでに 1 分かかる場合があります。

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 (Redis OSS) が `describe-serverless-caches` コマンドを使用してキャッシュの作成を完了したことを確認します。

```
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. 以下に示すように、アカウントが新しいロールを引き受けることができるIAM信頼ポリシードキュメントを作成します。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新しい 対応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: の Lambda 関数を作成する ElastiCache

ElastiCache キャッシュにアクセスするための Lambda 関数を作成するには、以下の手順を実行します。

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 関数のコード例を Python で提供します。

Python

次の Python コードの例では、 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。コードの`elasticache_endpoint`値を、前のステップ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 botocore.session
import redis
from botocore.model import ServiceId
from botocore.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 = botocore.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.use1.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 Valkey.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Valkey"
```

このコードは Python redis-py ライブラリを使用して項目をキャッシュに入れ、取得します。このコードは cachetools を使用して、生成された Auth IAM 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、次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で、フォルダパッケージを作成して redis-py ライブラリと cachetools ライブラリをインストールします。

```
mkdir package
```

pip を使用して redis-py 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、任意の zip ユーティリティを使用して、ルートに 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。

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

.zip ファイルに関数コードを追加します。Linux および macOS では、次のコマンドを実行します。Windows では、任意の zip ユーティリティを使用して、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ロールを作成する (実行ロール)

という名前の AWS マネージドポリシーをロールAWSLambdaVPCAccessExecutionRole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、create-function AWS CLI コマンドを使用して Lambda 関数 (AccessValkey) を作成します。

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

ロールオプションには、前のステップで作成した実行ロールARNの を使用します。vpc-config には、デフォルトの VPCのサブネットとデフォルトの VPCのセキュリティグループ ID のカンマ区切りリストを入力します。これらの値は Amazon VPCコンソールで確認できます。デフォルトの VPCのサブネットを検索するには、 VPCsを選択し、 AWS アカウントのデフォルトの を選択しますVPC。この のセキュリティグループを検索するにはVPC、Security に移動し、Security groups 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 AccessValkey  \
--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: で Lambda 関数をテストする ElastiCache

このステップでは、呼び出しコマンドを使用して Lambda 関数を手動で呼び出します。Lambda 関数を実行するUUIDと、 が生成され、Lambda コードで指定した ElastiCache キャッシュに書き込まれます。次に、Lambda 関数はキャッシュから項目を取得します。

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

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

2. Lambda 関数が正常に実行されたことを、次のように確認します。

   - output.txt ファイルを確認します。
   - CloudWatch コンソールを開き、関数のロググループ (/aws/lambda/) を選択して、CloudWatch Logs で結果を確認しますAccessValkey。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 Valkey.
     ```

   - AWS Lambda コンソールで結果を確認します。

ステップ 4: クリーンアップ (オプション)

クリーンアップするには、以下の手順を実行します。

ステップ 4.1: Lambda 関数を削除する

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

ステップ 4.2: サーバーレスキャッシュを削除する

キャッシュを削除します。

```
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を選択できます。ElastiCache では、クラスターの AWS アベイラビリティーゾーン間でノードタイプ、ノード数、ノード配置を選択して、ノードベースのクラスターを操作できます。 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、ソフトウェアパッチ適用を自動的に管理します。

セットアップに関する詳細については、「のセットアップ ElastiCache」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、「でのノードの管理 ElastiCache」を参照してください。独自の ElastiCache クラスターを設計する際の Amazon ElastiCache デプロイの主要なコンポーネントの概要については、以下の主要な概念を参照してください。

トピック

- ElastiCache コンポーネントと機能
- ElastiCache 用語
- チュートリアル: 独自のクラスターを設計する方法
- クラスターの削除
- その他の ElastiCache チュートリアルと動画
- でのノードの管理 ElastiCache
- でのクラスターの管理 ElastiCache
- Valkey、RedisOSS、および Memcached のセルフデザインキャッシュの比較
- Valkey または Redis のオンライン移行 OSS
- のリージョンとアベイラビリティーゾーンの選択 ElastiCache

ElastiCache コンポーネントと機能

以下は、Amazon ElastiCache デプロイの主要なコンポーネントの概要です。

トピック

- ElastiCache ノード
- ElastiCache シャード
- ElastiCache クラスター
- ElastiCache レプリケーション

- ElastiCache エンドポイント
- ElastiCache パラメータグループ
- ElastiCache セキュリティ
- ElastiCache サブネットグループ
- ElastiCache バックアップ
- ElastiCache イベント

ElastiCache ノード

ノードは ElastiCache デプロイ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接続された の固定サイズのチャンクですRAM。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、「スケーリング ElastiCache」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には、独自のドメインネームサービス (DNS) 名とポートがあります。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。

ノードは、ノードの使用に対してのみ支払うベースで pay-as-you-go購入できます。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、ノードを追加する必要がある時間帯のノードを購入 pay-as-you-goできます。リザーブドノードの詳細については、「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、「でのノードの管理 ElastiCache」を参照してください。

ElastiCache シャード

Valkey または Redis OSS シャード (APIおよび のノードグループと呼ばれますCLI) は、1 ~ 6 個の関連ノードのグループです。クラスターモードが有効になっている Valkey または Redis OSS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、速く、より簡単に管理できる部分に分割するデータベースパーティショニング方法です。これにより、複数の別々のセクションにオペレーションを分散することで、データベースの効率を高めることができます。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、多くの利点が得られます。

クラスターモードが有効になっている Valkey クラスターと Redis OSSクラスターは、最大 500 個のシャードを持つことができ、データはシャード間でパーティション分割されます。Valkey または Redis OSS エンジンのバージョンが 5.0.6 以上であれば、ノードまたはシャードの制限をクラスターごとに最大 500 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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることが含まれます。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、「でのシャードの使用 ElastiCache」を参照してください。

ElastiCache クラスター

クラスターは、1 つ以上のノード の論理グループです。データは、Memcached クラスター内のノード間、およびクラスターモードが有効になっている Valkey または Redis OSSクラスター内のシャード間でパーティション化されます。

多くの ElastiCache オペレーション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。

- でのクラスターの管理 ElastiCache および でのノードの管理 ElastiCache

  クラスター、ノードおよび関連オペレーションに関する情報。
- AWS サービス制限: Amazon ElastiCache

  ノードやクラスターの最大数など、 ElastiCache 制限に関する情報。これらの制限の一部を超えるには、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

  クラスターと Valkey または Redis OSSレプリケーショングループの耐障害性の向上に関する情報。

一般的なクラスターの設定

以下は一般的なクラスターの構成です。

Valkey または Redis OSSクラスター

クラスターモードが無効になっている Valkey または Redis OSSクラスターには、常に 1 つのシャード (APIおよび では CLI1 つのノードグループ) のみが含まれます。Valkey または Redis OSS シャードには 1 ~ 6 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。

耐障害性を向上させるには、Valkey または Redis OSSクラスターに少なくとも 2 つのノードがあり、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。

Valkey または Redis OSSクラスターの需要の変化に応じて、スケールアップまたはスケールダウンできます。これを行うには、クラスターを別のノードインスタンスタイプに移動します。アプリケーションが読み取り集約型の場合は、読み取り専用レプリカをクラスターに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、「のデータ階層化 ElastiCache」を参照してください。

ElastiCache では、Valkey または Redis OSSクラスター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できます。スケールアップ/ダウンの詳細については、「Valkey または Redis の単一ノードクラスターのスケーリング OSS (クラスターモードが無効)」または「Valkey または Redis のレプリカノードのスケーリング OSS (クラスターモードが無効)」を参照してください。

Memcached の一般的なクラスター設定

Memcached は、各 AWS クラスターに 1～60 ノードを持つリージョンごとに、顧客ごとに最大 300 ノードをサポートします。Memcached クラスターのノードにデータを分割することができます。

Memcached エンジンを実行すると、クラスターは 1～60 ノードで構成されます。データベースをノード間で分割できます。アプリケーションによって各ノードのエンドポイントに対して読み書きされます。詳細については、「自動検出」を参照してください。

耐障害性を向上させるには、クラスターの AWS リージョン内のさまざまなアベイラビリティーゾーン (AZs) に Memcached ノードを見つけます。この方法により、1 つのアベイラビリティーゾーンで発生した障害がクラスター全体とアプリケーションに与える影響を最小限にできます。詳細については、「障害の軽減」を参照してください。

Memcached クラスターの需要の変化に合わせて、ノードの追加や削除で規模を拡大したり縮小したりできます。また、新しいノードにまたがってデータを再分割できます。データを分割するときは、整合性のあるハッシュを使用することをお勧めします。整合性のあるハッシュの詳細については、「効率的なロードバランシングのための ElastiCache クライアントの設定 (Memcached)」を参照してください。

ElastiCache レプリケーション

Valkey と Redis の場合OSS、レプリケーションはシャード内の 2 つから 6 つのノード (ノードグループCLIと呼ばれる APIおよび) にグループ化す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。レプリケーションは Valkey と Redis ElastiCache でのみ使用できOSS、 ElastiCache (Memcached) では利用できません。

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ー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。

Valkey クラスターまたは Redis OSSクラスターは、1 つのシャード (ノードグループ CLIと呼ばれるAPIおよび) をサポートします。

API および 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が使用されますが、結果は同じです。次の表は、レプリケーションを実装するための API と CLIの条件を示しています。

レプリケーションの比較: Valkey または Redis OSS (クラスターモードが無効) および Valkey または Redis OSS (クラスターモードが有効)--> Valkey または Redis OSSクラスターとクラスターモードが無効の Valkey または Redis OSSクラスターの比較

次の表では、Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループと Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループの特徴の比較を示します。

	OSS クラスターモードが無効になっている Valkey または Redis クラスター	OSS クラスターモードが有効になっている Valkey または Redis クラスター
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	不可	可能
レプリカの追加/削除	あり	可能
ノードグループの追加/削除	不可	可能
サポートの拡大	あり	可能
エンジンアップグレードのサポート	あり	可能

	OSS クラスターモードが無効になっている Valkey または Redis クラスター	OSS クラスターモードが有効になっている Valkey または Redis クラスター
レプリカをプライマリに昇格	可能	自動
マルチ AZ	オプションです。	必須
バックアップ/復元	あり	可能

注意:

どのプライマリにもレプリカがなく、プライマリに障害が発生した場合、そのプライマリ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、Valkey または Redis OSS (クラスターモードが有効) に移行できます。

バックアップと復元を使用して、Valkey または Redis OSS (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 および CLI、ノードグループ内) 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、個々のノードは、その AWS リージョン内の複数のアベイラビリティーゾーンにプロビジョニングできます。

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ーゾーンに 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 OSS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、信頼性が高く、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、 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 ElastiCache 、 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。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。

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。

ElastiCache エンドポイント

エンドポイントは、アプリケーションが ElastiCache ノードまたはクラスターへの接続に使用する一意のアドレスです。

クラスターモードが無効OSSになっている Valkey または Redis の単一ノードエンドポイント

単一ノードの Valkey クラスターまたは Redis OSSクラスターのエンドポイントは、読み取りと書き込みの両方でクラスターに接続するために使用されます。

クラスターモードが無効OSSになっている Valkey または Redis のマルチノードエンドポイント

クラスターモードが無効になっている複数のノードの Valkey または Redis OSS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を読み取りオペレーションに使用します (API/CLI では、これらはリードエンドポイントと呼ばれます) 。

Valkey または Redis OSS (クラスターモードが有効) エンドポイント

OSS クラスターモードが有効になっている Valkey または Redis 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、「での接続エンドポイントの検索 ElastiCache」を参照してください。

ElastiCache (Memcached) エンドポイント

Memcached クラスターの各ノードには、独自のエンドポイントがあります。クラスターには、設定エンドポイントと呼ばれるエンドポイントもあります。自動検出を有効にして設定エンドポイントに

接続した場合、クラスターからノードの追加や削除を行った後であっても、アプリケーションは自動的に各ノードエンドポイントを検出します。詳細については、「自動検出」を参照してください。

詳細については、「 エンドポイント」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の名前付きコレクションです。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、および変更できるパラメータについては、「」を参照してください。DescribeEngineDefaultParameters (CLI: describe-engine-default-parameters).

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 ElastiCache パラメータの設定。

ElastiCache セキュリティ

セキュリティを強化するために、 ElastiCache ノードへのアクセスは、許可する 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、クラスターにアクセスできる Amazon EC2インスタンスを制御できます。

デフォルトでは、すべての新しい ElastiCache クラスターは 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実行されている Amazon EC2インスタンスからクラスターアクセスを許可できます。

は、ノードアクセスの制限に加えて、 の指定されたバージョンを実行しているノードの暗号化 ElastiCache をサポートしTLS、インプレース暗号化をサポートしています ElastiCache。詳細については、次を参照してください。

- Amazon のデータセキュリティ ElastiCache
- Valkey および Redis OSS 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環境で実行されているクラスターに指定できるサブネット (通常はプライベート) のコレクションです。

Amazon でクラスターを作成する場合はVPC、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 は、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内のサブネットと 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状況の詳細については、以下を参照してください。

- Amazon VPCs と ElastiCache セキュリティ
- ステップ 3. クラスターへのアクセスを許可する
- サブネットおよびサブネットグループ

ElastiCache バックアップ

バックアップは、 point-in-timeValkey または Redis OSSクラスター、サーバーレスキャッシュ、または Memcached サーバーレスキャッシュのコピーです。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OSS実行されている Valkey または Redis のバージョンに応じて、バックアッププロセスを成功させるには、予約済みメモリの量が異なります。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
- Valkey または Redis OSS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を確認する

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、 は特定の Amazon SNSトピックに通知 ElastiCache 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 ElastiCache イベントの Amazon SNSモニタリング。

ElastiCache 用語

2016 年 10 月、Amazon は Redis 3.2 OSS のサポート ElastiCache を開始しました。その時点で、最大 500 個のシャード (ElastiCache APIおよび ではノードグループと呼ばれます AWS CLI) にデータを分割するサポートが追加されました。以前のバージョンとの互換性を維持するために、APIバージョン 2015-02-02 オペレーションを拡張して、新しい Redis OSS機能を追加しました。

同時に、この新しい機能で使用され、業界全体で一般的な ElastiCache コンソールで用語の使用を開始しました。これらの変更は、ある時点で、 API および で使用される用語がコンソールで使用される用語とは異なるCLI可能性があることを意味します。次のリストは、 API CLIと コンソールで異なる可能性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、 one-to-oneノードとキャッシュクラスターの間には関係があります。したがって、 ElastiCache コンソールはしばしばこの用語を互換的に使用していました。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。

と ElastiCache APIは、これまでと同様に用語 AWS CLI を引き続き使用します。

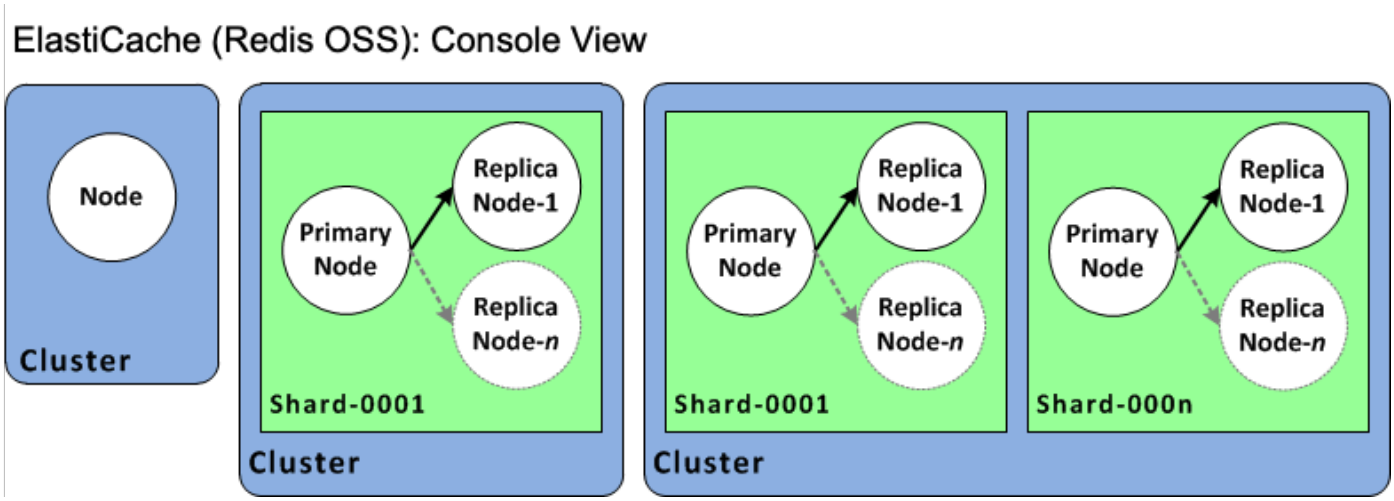
クラスターと Valkey または Redis OSSレプリケーショングループ

コンソールは、すべての ElastiCache (Redis OSS) クラスターに クラスターという用語を使用するようになりました。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。

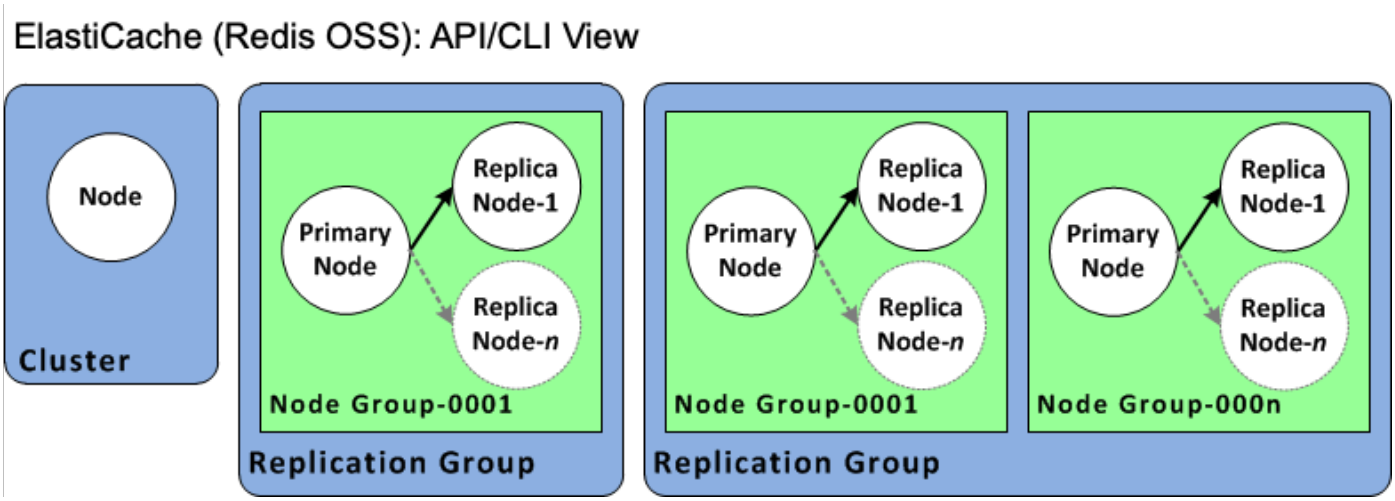
- クラスターが単一ノードの Valkey または Redis OSSクラスターの場合。
- クラスターが単一のシャード OSS (ノードグループ CLIと呼ばれる APIおよび) 内のレプリケーションをサポートする Valkey または Redis (クラスターモードが無効) クラスターである場合。
- クラスターが 1 ~ 90 シャード以内、または制限引き上げリクエストで最大 500 までのレプリケーションをサポートする Valkey または Redis OSS (クラスターモードが有効) クラスターである場合。この制限の拡大をリクエストするには、「AWS のサービスの制限」を参照し、制限タイプとして [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。

Valkey または Redis OSSレプリケーショングループの詳細については、「」を参照してくださいレプリケーショングループを使用する高可用性。

次の図は、コンソールの観点から見た ElastiCache (Redis OSS) クラスターのさまざまなトポロジを示しています。



および ElastiCache API AWS CLI オペレーションでは、単一ノード ElastiCache (Redis OSS) クラスターをマルチノードの Valkey または Redis OSSレプリケーショングループと区別します。次の図は、 および AWS CLI の観点からのさまざまな ElastiCache (Redis OSS) トポロジを示しています ElastiCache API。



Valkey または Redis OSS レプリケーショングループとグローバルデータストアの比較

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コレクションです。一方、Valkey または Redis OSS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になっている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 リージョンにある必要があります。

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。

チュートリアル: 独自のクラスターを設計する方法

Valkey と Redis 用に独自のクラスターを設計する方法は次のとおりですOSS。

トピック

独自の ElastiCache （ バルキー) クラスターの設計

ElastiCache （ バルキー) クラスターの設計を開始するために実行する必要がある 1 回限りのアクションを次に示します。

ステップ 1: サブネットグループの作成

ElastiCache （ バルキー) 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、 内のキャッシュクラスターに指定するサブネットのコレクションですVPC。でキャッシュクラスターを起動するときはVPC、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、そのキャッシュサブネットグループ ElastiCache を使用して、そのサブネット内の 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。

設定の詳細については、 ElastiCache 「」を参照してくださいのセットアップ ElastiCache。

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

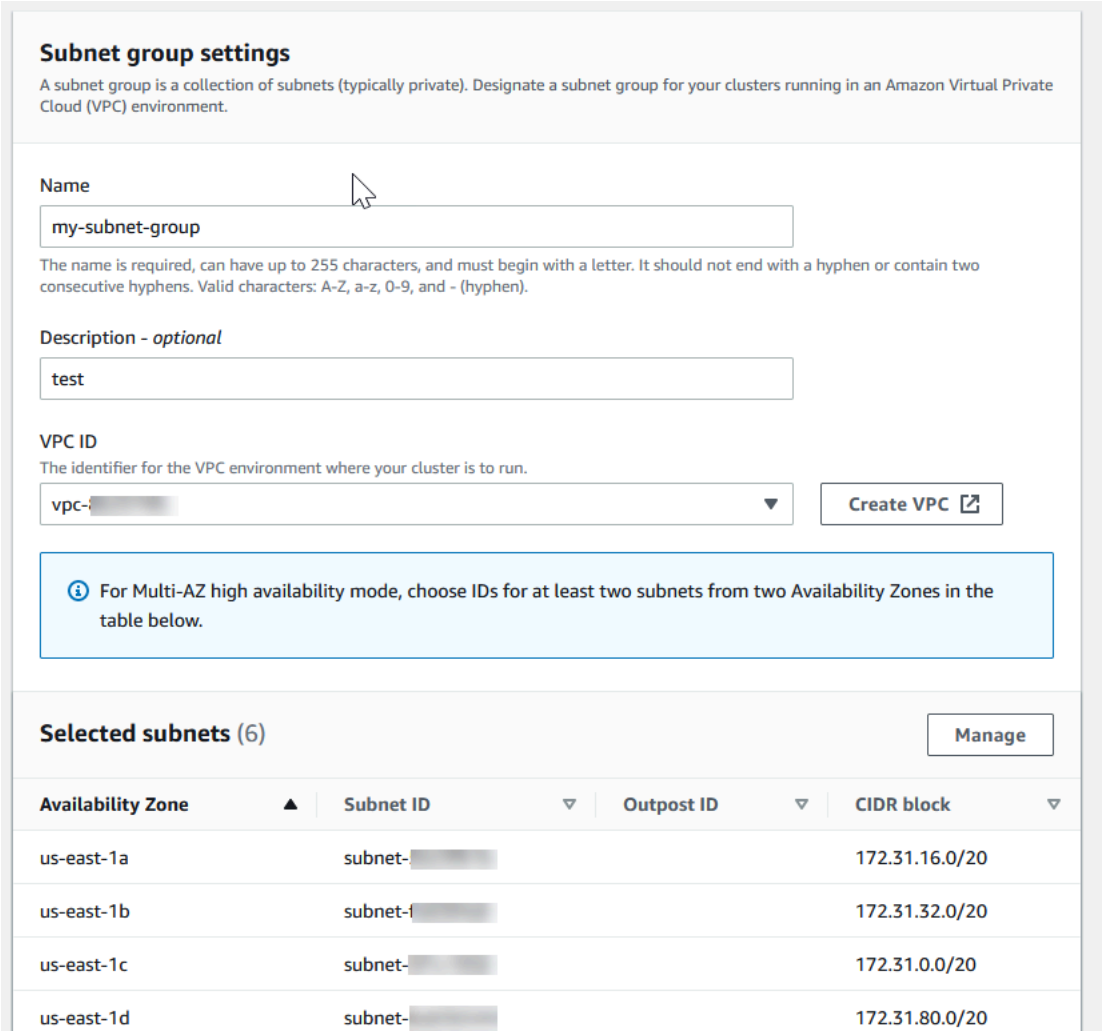
1. AWS マネジメントコンソールにサインインし、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションリストで [Subnet Groups] を選択します。
3. Create Subnet Group を選択します。
4. Create Subnet Group ウィザードで、次の操作を行います。すべての設定が正しいことを確認したら、[Yes, Create] を選択します。

    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 ボックスに、VPC作成した Amazon を選択します。

    d. アベイラビリティーゾーンとサブネット ID リストで、プライベートサブネットのアベイラビリティーゾーンまたは での ローカルゾーンの使用 ElastiCache と ID を選択し、 の追加を選択します。



5. 表示された確認メッセージで、Close を選択します。

新しいサブネットグループは、 ElastiCache コンソールのサブネットグループリストに表示されます。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 「」トピックを参照してください。create-cache-subnet-group.

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、でのクラスターの準備 ElastiCache 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。削除されるまで、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 使用率の詳細については、「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仮想プライベートクラウド (VPC) で起動されます。

Valkey (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 Valkey (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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 右上隅のリストから、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. 「仮想VPCプライベートクラウドの作成」 (VPC) で説明されているステップに従って作成「」を選択します。
5. ElastiCache ダッシュボードページで、Valkey キャッシュまたは Redis OSSキャッシュ を選択し、Valkey キャッシュの作成 または Redis OSSキャッシュ 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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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、「Valkey パラメータと Redis

OSSパラメータ」および「ElastiCache パラメータグループの作成」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、「のデータ階層化 ElastiCache」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 の場合、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ために IPv4と IPv6 アドレスの両方が割り当てられているデュアルスタックサブネットと、 IPv6IPv6のみのサブネットが動作するように割り当てられているデュアルスタック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、それが属する 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。

- でネットワークタイプを選択する ElastiCache.
- にサブネットを作成しますVPC。

での ローカルゾーンの使用 ElastiCache 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] (アベイラビリティーゾーンの配置) には 2 つのオプションがあります。

   - 設定なし – アベイラビリティーゾーン ElastiCache を選択します。
   - [アベイラビリティーゾーンの指定] – 各クラスターに対するアベイラビリティーゾーンを指定します。

     アベイラビリティー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ーゾーンを選択します。

   詳細については、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」を参照してください。

5. [Next] (次へ) を選択します。
6. 詳細バルキーまたは Redis OSS設定の下

   - [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、「 の AWS カスタマーマネージドキーの使用KMS」を参照してください。

- [転送時の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「転送中の暗号化」を参照してください。Valkey および Redis OSS 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、次のいずれかのアクセスコントロールオプションを指定するように求められます。
  - 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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。
  - AUTH デフォルトユーザー – Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。
- AUTH — Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

> **Note**
>
> Valkey および 3.2.6 以降の Redis OSSバージョンでは、バージョン 3.2.10 を除き、Redis が唯一のオプションOSSAUTHです。

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。のデフォルトのセキュリティグループを使用するVPCか、新しいセキュリティグループを作成できます。

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

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

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、クラスターのシステムメンテナンスをスケジュールする ElastiCache毎週の時間で、通常は 1 時間です。は ElastiCache、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。すべての時間は UCT 時間です。

   詳細については、「ElastiCache クラスター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]:

   - ログ形式 で、テキストまたは を選択しますJSON。
   - 送信先タイプ で、CloudWatch ログ または Kinesis Firehose のいずれかを選択します。
   - Log destination で、Create new を選択して Logs CloudWatch ロググループ名または Firehose ストリーム名を入力するか、Select existing を選択して Logs CloudWatch 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 リソースの管理に役立つように、タグの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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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使用可能になるとすぐに、クラスターEC2へのアクセスを Amazon に許可し、クラスターに接続して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を許可する」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。「でのクラスターの削除 ElastiCache」を参照してください。

Valkey (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コードは、レプリカのない Valkey (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 valkey \
--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 valkey ^
--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、「Valkey または Redis OSS (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Valkey または Redis OSS (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を許可する

このセクションでは、Amazon EC2インスタンスの起動と接続に精通し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) 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す。これは、この演習でも同様です。

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前に、EC2インスタンスがクラスターにアクセスすることを承認する必要があります。

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 のクラスターに接続する必要がある場合ですVPC。同じ 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最も簡単な方法はVPC、以下を実行することです。

1. クラスターVPC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。例えば、このセキュリティグループのカスタムルールを作成して、クラスターの作成時にクラスターに割り当てたポートと、クラスターTCPへのアクセスに使用する IP アドレスを使用してアクセスを許可できます。

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

   > **Important**
   >
   > Amazon ElastiCache セキュリティグループは、Amazon Virtual Private Cloud 環境 () で実行されていないクラスターにのみ適用されますVPC。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。
   > Amazon で ElastiCache ノードを実行している場合はVPC、 ElastiCache セキュリティグループとは異なる Amazon VPC セキュリティグループを使用してクラスターへのアクセスを制御します。Amazon ElastiCache での の使用の詳細についてはVPC、「」を参照してください。 Amazon VPCs と ElastiCache セキュリティ

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) VPC 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、このセキュリティグループのルールを設定して、ポート 22 経由でEC2インスタンスTCPへのアクセスを許可できます。

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カスタムルールをクラスターのセキュリティグループに作成します。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はサブネットを他のアベイラビリティーゾーンの任意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。

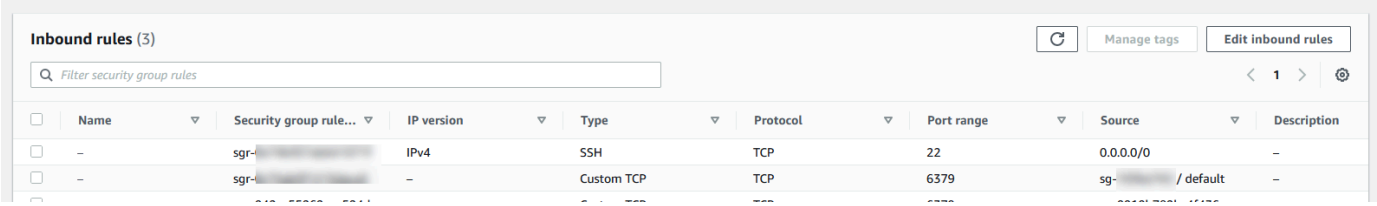
別のVPCセキュリティグループからの接続を許可するルールをセキュリティグループに作成するには

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、[Security Groups] 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. タイプ からカスタムTCPルール 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。

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

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



アクセスを有効にしたので、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 、別の AWS リージョンVPC、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方法については、以下を参照してください。

- Amazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC
- 外部から ElastiCache リソースにアクセスする AWS

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を許可する」を完了します。

このセクションでは、Amazon EC2インスタンスを作成し、接続できることを前提としています。これを行う方法については、「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、クラスターノードへの接続を許可した場合にのみ接続できます。

ノードのエンドポイントを見つける

クラスターが使用可能な状態で、クラスターへのアクセスを許可したら、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。

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

Valkey (クラスターモードが無効) クラスターに 1 つのノードしかない場合、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、クラスター内のプライマリノードに常に解決されるDNS名前です。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、 ElastiCache 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を均等に分割します。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 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。

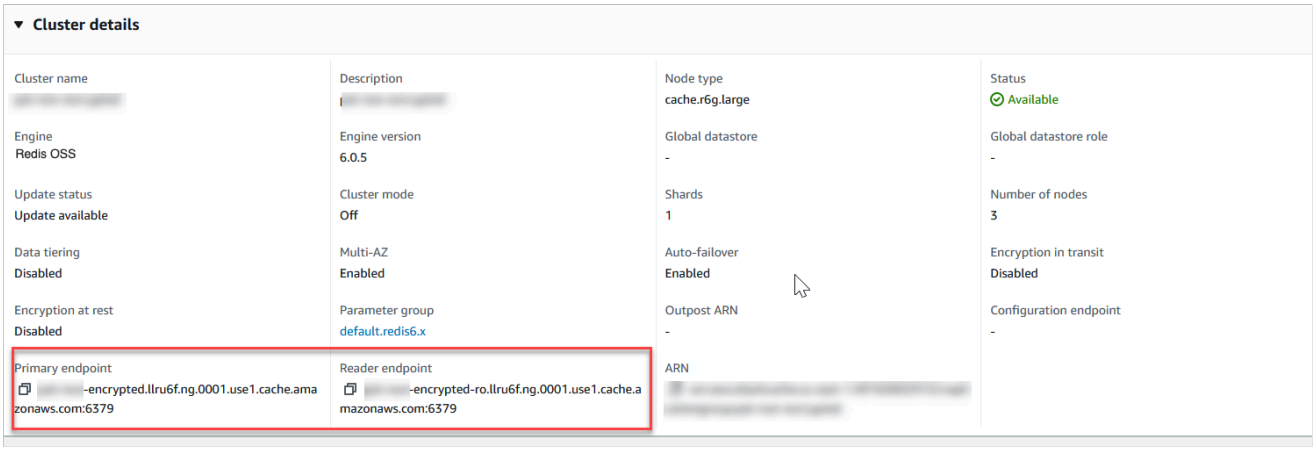
> **Note**
>
> リーダーエンドポイントはロードバランサーではありません。これは、レプリカノードの 1 つの IP アドレスにラウンドロビン方式で解決されるDNSレコードです。

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。

Valkey (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. 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から、バルキーキャッシュ または Redis OSSキャッシュ を選択します。

   クラスター画面には、既存の Valkey または Redis OSSサーバーレスキャッシュ、Valkey (クラスターモードが無効) 、および Valkey (クラスターモードが有効) クラスターを含むリストが表示されます。Valkey (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。

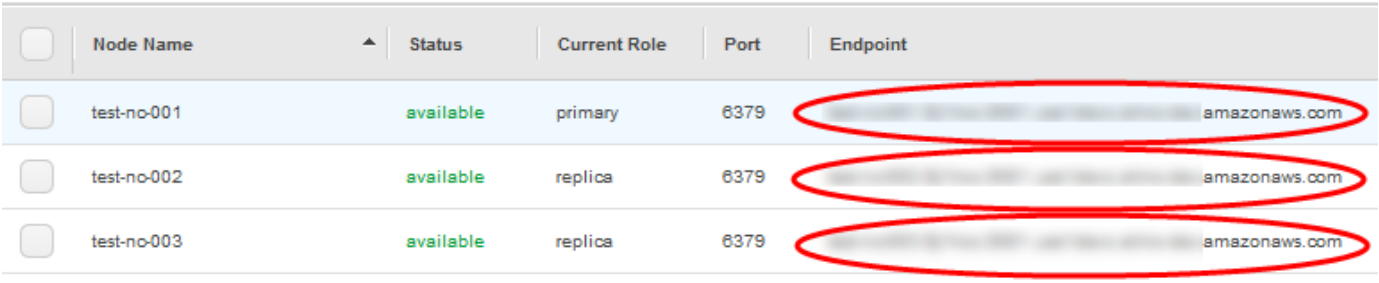


Valkey (クラスターモードが無効) クラスターのプライマリエンドポイントとリーダーエンドポイント

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. Valkey (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、クラスター名を選択し、Nodes タブを選択することで、クラスターのレプリカノードエンドポイントを見つけることができます。

   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。



   Valkey (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:

   a. 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。

   b. 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。

Valkey (クラスターモードが無効) のプライマリエンドポイントは次のようになります。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
```

エンドポイントを見つける方法の詳細については、実行中のエンジンとクラスターの該当するトピックを参照してください。

- での接続エンドポイントの検索 ElastiCache
- Valkey または Redis OSS (クラスターモードが有効) クラスターのエンドポイントの検出 (コンソール) —クラスターの設定エンドポイントが必要です。
- エンドポイントの検索 (AWS CLI)
- エンドポイントの検出 (ElastiCache API)

Valkey または Redis OSS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)

必要なエンドポイントができたので、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、valkey-cli ユーティリティを使用してクラスターに接続します。最新バージョンの valkey-cli は、暗号化/認証が有効になっているクラスターを接続する SSL/TLS もサポートしています。

次の例では、Amazon Linux と Amazon Linux 2 を実行する 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 valkey-cli をインストールしてコンパイルする方法については、特定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、計画外使用のみを目的として valkey-cli ユーティリティを使用して接続をテストします。サポートされている Valkey および Redis OSSクライアントのリストについては、「 Valkey ドキュメント」を参照してください。で を使用する例については ElastiCache、 AWS SDKs「」を参照してくださいチュートリアル: Python と の開始方法 ElastiCache。

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、 を置き換えます。*primary-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey または Redis のデフォルトポートOSSは 6379 です) 。

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

   Valkey または Redis OSS コマンドプロンプトの結果は次のようになります。

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

2. Valkey コマンドまたは Redis OSS 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、 を置き換えます。*configuration-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey または Redis のデフォルトポートOSSは 6379 です) 。

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

> **Note**
>
> 前のコマンドでは、オプション -c は - ASKおよび -MOVED リダイレクト の後にクラスターモードを有効にします。

Valkey または Redis OSS コマンドプロンプトの結果は次のようになります。

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

2. Valkey コマンドまたは Redis OSS 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になっていない場合、 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 OSS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、valkey-cli は Valkey または 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OSS。オプションは、前のコマンドラインアクセスをダウンロードしてセットアップするセクションに示すように、valkey-cli コンパイル時に SSL/TLS BUILD_TLS=yesを有効にします。有効化AUTHはオプションです。ただし、 を有効にするには、転送中の暗号化を有効にする必要がありますAUTH。暗号化と認証の詳細については ElastiCache 、「」を参照してくださいElastiCache 転送中の暗号化 (TLS) 。

> **Note**
>
> valkey-cli --tlsで オプションを使用して、クラスターモードが有効と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、オプションを使用してAUTHパスワード-aを指定できます。

次の例では、必ず を置き換えてください。*cluster-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey または Redis のデフォルトポートOSS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/valkey-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/valkey-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/valkey-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/valkey-cli -c -h cluster-endpoint --tls -p port number
```

クラスターに接続したら、前述の例に示すように、暗号化されていないクラスターに対して Valkey コマンドまたは Redis OSS コマンドを実行できます。

valkey-cli 代替

クラスターがクラスターモードを有効にしておらず、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、valkey-cli コンパイルを経由しない場合、telnet または openssl を使用できます。次のコマンド例では、必ず を置き換えてください。*cluster-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey または Redis のデフォルトポートOSS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* クラスターのパスワードを使用します。

```
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
```

Valkey または Redis OSS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)

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

次の例では、valkey-cli ユーティリティを使用して、暗号化が有効になっていず、Valkey または Redis を実行しているクラスターに接続しますOSS。Valkey または Redis OSSと使用可能なコマンドの詳細については、Valkey ウェブサイトの「Valkey コマンドと Redis OSS コマンド」を参照してください。

valkey-cli を使用して暗号化が有効になっていない Valkey または Redis OSS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. インターネットブラウザhttps://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zipにリンクをコピーして貼り付け、 で利用可能なリリースから Valkey クライアントの zip ファイルをダウンロードします。 GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504

   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、Valkey ディレクトリに変更して、コマンド を実行しますc:\Valkey>valkey-cli -h *Redis_Cluster_Endpoint* -p 6379。

   例:

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

3. Valkey または Redis OSS コマンドを実行します。

   これでクラスターに接続され、次のような Valkey コマンドまたは Redis OSS 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 valkey-cli
```

次のステップ

入門演習を試したので、以下のセクションを参照して、 ElastiCache および利用可能なツールの詳細を確認できます。

- の開始方法 AWS
- Amazon Web Services のツール
- AWS コマンドラインインターフェイス
- Amazon ElastiCache API リファレンス

入門演習を完了したら、以下のセクションを読んで ElastiCache 管理の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。

独自の ElastiCache (Redis OSS) クラスターの設計

以下は、独自の ElastiCache (Redis OSS) クラスターを設計するために実行する必要がある 1 回限りのアクションです。

設定の詳細については、 ElastiCache 「」を参照してくださいのセットアップ ElastiCache。

トピック

ステップ 1: サブネットグループの作成

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、 内のキャッシュクラスターに指定するサブネットのコレクションですVPC。でキャッシュクラスターを起動するときはVPC、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、そのキャッシュサブネットグループ ElastiCache を使用して、そのサブネット内の 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。

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

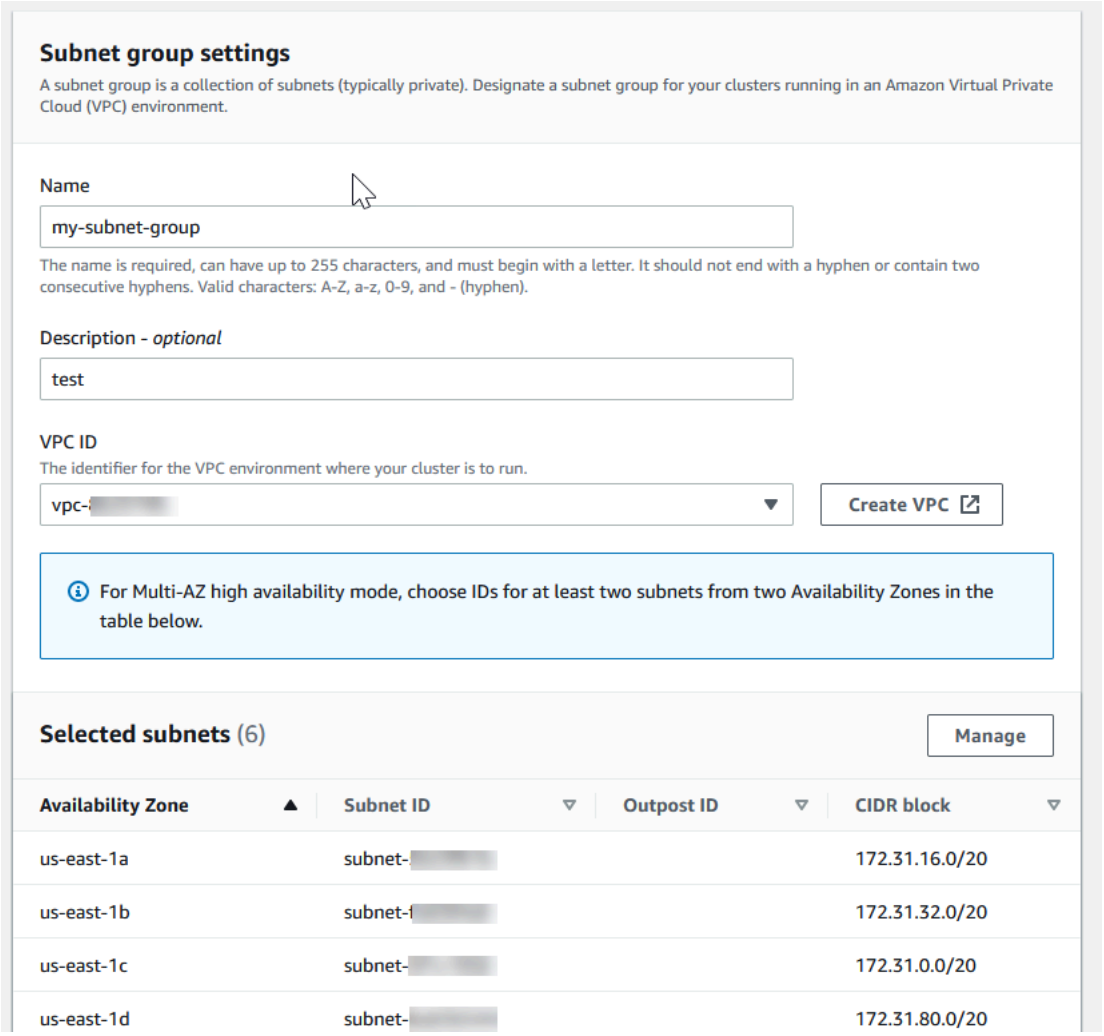
1. AWS マネジメントコンソールにサインインし、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションリストで [Subnet Groups] を選択します。
3. Create Subnet Group を選択します。
4. Create Subnet Group ウィザードで、次の操作を行います。すべての設定が正しいことを確認したら、[Yes, Create] を選択します。

    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 ボックスに、VPC作成した Amazon 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アベイラビリティーゾーンまたは [Local Zone] と ID を選択し、[Add] を選択します。



5. 表示された確認メッセージで、Close を選択します。

新しいサブネットグループは、 ElastiCache コンソールのサブネットグループリストに表示されます。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 「」トピックを参照してください。create-cache-subnet-group.

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、でのクラスターの準備 ElastiCache 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。削除されるまで、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 使用率の詳細については、「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仮想プライベートクラウド (VPC) で起動されます。

Redis OSS (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 OSS (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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 右上隅のリストから、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. Create VPC を選択し、「仮想プライベートクラウドの作成 (VPC)」で説明されているステップに従います。
5. ElastiCache ダッシュボードページで、Valkey キャッシュ または Redis OSSキャッシュ を選択します。この演習では、Redis OSSキャッシュ を選択し、Redis OSS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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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、「Valkey パラメータと Redis OSSパラメータ」および「 ElastiCache パラメータグループの作成」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、「のデータ階層化 ElastiCache」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 では、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ために IPv4と IPv6 アドレスの両方が割り当てられているデュアルスタックサブネットと、 IPv6IPv6のみのサブネットが動作するように割り当てられているデュアルスタック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、それが属する 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。

- でネットワークタイプを選択する ElastiCache.
- にサブネットを作成しますVPC。

での ローカルゾーンの使用 ElastiCache 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] (アベイラビリティーゾーンの配置) には 2 つのオプションがあります。

   - 設定なし – アベイラビリティーゾーン ElastiCache を選択します。
   - [アベイラビリティーゾーンの指定] – 各クラスターに対するアベイラビリティーゾーンを指定します。

     アベイラビリティー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ーゾーンを選択します。

   詳細については、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」を参照してください。

5. [Next] (次へ) を選択します。
6. 詳細 Redis OSS設定の下

   - [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、「 の AWS カスタマーマネージドキーの使用KMS」を参照してください。

- [転送時の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「転送中の暗号化」を参照してください。Redis OSS 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、次のいずれかのアクセスコントロールオプションを指定するように求められます。
  - 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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。
  - AUTH デフォルトユーザー – Valkey および Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。
- AUTH — Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

> **Note**
>
> OSS バージョン 3.2.10 を除く 3.2.6 以降の Redis バージョンでは、Redis が唯一のオプションOSSAUTHです。

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。のデフォルトのセキュリティグループを使用するVPCか、新しいセキュリティグループを作成できます。

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

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。

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

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、クラスターのシステムメンテナンスをスケジュールする ElastiCache毎週の時間、通常は 1 時間です。は ElastiCache、メンテナンスウィンドウの日時を選択する (設定なし) ことも、日、時刻、期間を自分で選択することもできます (メンテナンスウィンドウを指定)。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。すべての時間は UCT 時間です。

   詳細については、「 ElastiCache クラスター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]:

   - ログ形式 で、テキストまたは を選択しますJSON。
   - 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
   - Log destination で、Create new を選択して Logs CloudWatch ロググループ名または Firehose ストリーム名を入力するか、Select existing を選択して Logs CloudWatch 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 リソースの管理に役立つように、タグの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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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利用可能になるとすぐに、クラスターへのアクセスを Amazon に許可し、クラスターに接続して使用を開始できます。 EC2詳細については、「ステップ 3. クラスターへのアクセスを許可する」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。「でのクラスターの削除 ElastiCache」を参照してください。

Redis OSS (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コードは、レプリカなしで Redis OSS (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、「Valkey または Redis OSS (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Valkey または Redis OSS (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精通し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) 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す。これは、この演習の場合も同様です。

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前に、EC2インスタンスがクラスターにアクセスすることを承認する必要があります。必要な手順は、クラスターを EC2-VPC または EC2-Classic に起動したかどうかによって異なります。

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 のクラスターに接続する必要がある場合ですVPC。同じ 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最も簡単な方法はVPC、以下を実行することです。

1. クラスターVPC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。例えば、このセキュリティグループのカスタムルールを作成して、クラスターの作成時にクラスターに割り当てたポートと、クラスターTCPへのアクセスに使用する IP アドレスを使用してアクセスを許可できます。

   Redis OSS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 です6379。

   > **Important**
   >
   > Amazon ElastiCache セキュリティグループは、Amazon Virtual Private Cloud 環境 () で実行されていないクラスターにのみ適用されますVPC。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。
   > Amazon で ElastiCache ノードを実行している場合はVPC、 ElastiCache セキュリティグループとは異なる Amazon VPC セキュリティグループを使用してクラスターへのアクセスを制御します。Amazon ElastiCache での の使用の詳細についてはVPC、「」を参照してください。 Amazon VPCs と ElastiCache セキュリティ

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) VPC 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、このセキュリティグループのルールを設定して、ポート 22 経由でEC2インスタンスTCPへのアクセスを許可できます。

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カスタムルールをクラスターのセキュリティグループに作成します。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**
>
> を使用する予定がある場合はでのローカルゾーンの使用 ElastiCache、有効にしていることを確認してください。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はサブネットを他のアベイラビリティーゾーンの任意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。

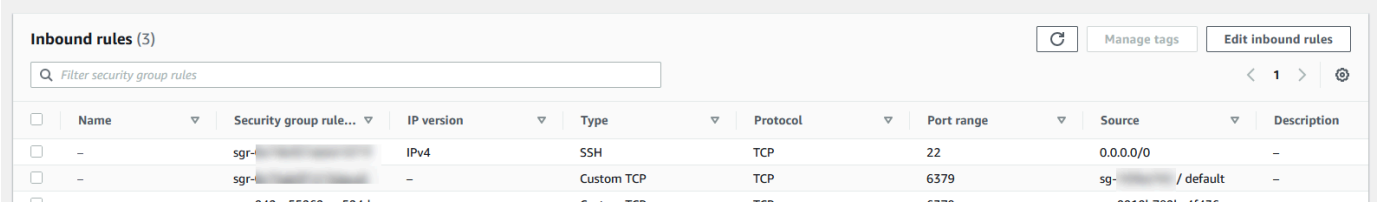
別のVPCセキュリティグループからの接続を許可するルールをセキュリティグループに作成するには

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、[Security Groups] 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. タイプ からカスタムTCPルール 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 OSS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 です6379。

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



アクセスを有効にしたので、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 、別の AWS リージョンVPC、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方法については、以下を参照してください。

- Amazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC
- 外部から ElastiCache リソースにアクセスする AWS

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インスタンスを作成し、接続できることを前提としています。これを行う方法については、「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、クラスターノードへの接続を許可した場合にのみ接続できます。

ノードのエンドポイントを見つける

クラスターが使用可能な状態で、そのクラスターへのアクセスを許可したら、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。

Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)

Redis OSS (クラスターモードが無効) クラスターに 1 つのノードしかない場合、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、クラスター内のプライマリノードに常に解決されるDNS名前です。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、 ElastiCache (Redis OSS) 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受信接続を均等に分割します。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 (Redis OSS) 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。

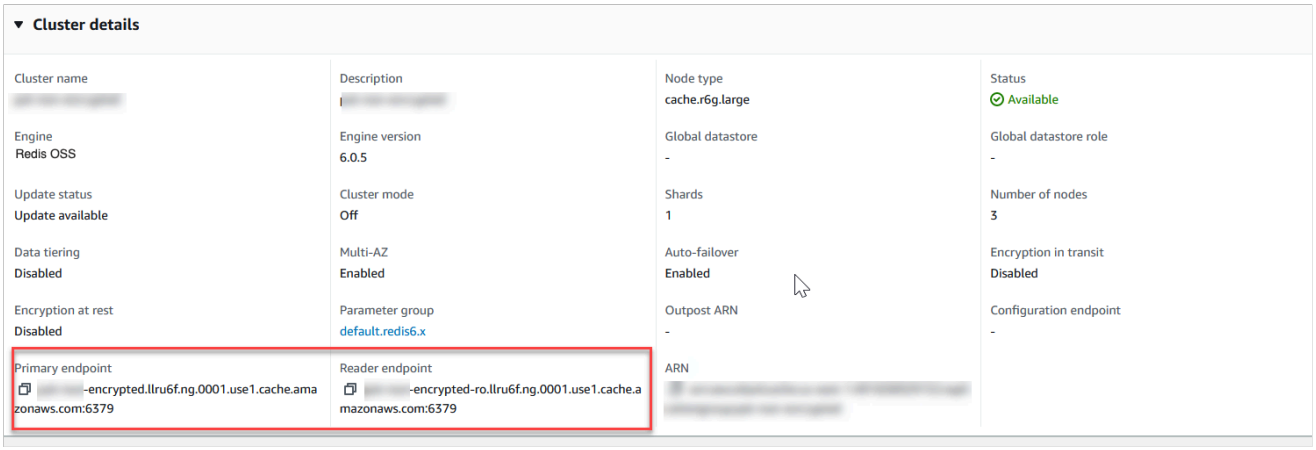
> **Note**
>
> リーダーエンドポイントはロードバランサーではありません。これは、レプリカノードの 1 つの IP アドレスにラウンドロビン方式で解決されるDNSレコードです。

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 OSS (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. 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から、Redis OSSキャッシュ を選択します。

   クラスター画面には、既存の Valkey または Redis OSSサーバーレスキャッシュ、Redis OSS (クラスターモードが無効) クラスター、Redis OSS (クラスターモードが有効) クラスターを含むリストが表示されます。Redis OSS (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。

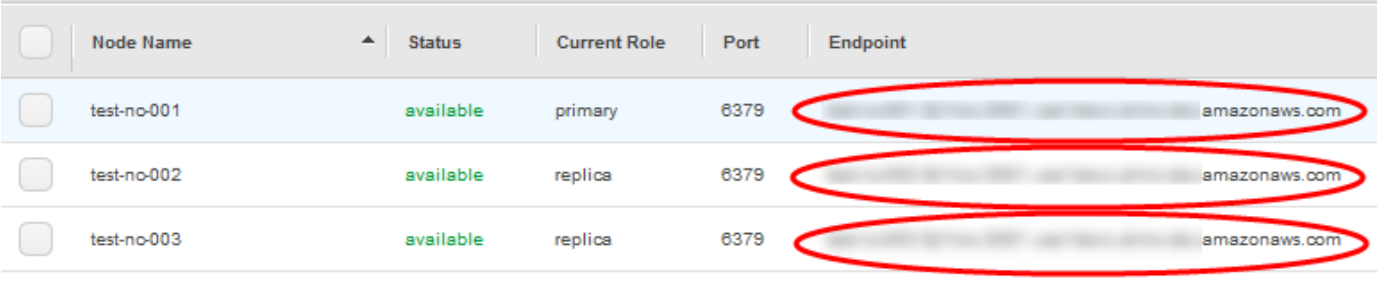


Redis OSS (クラスターモードが無効) クラスターのプライマリエンドポイントとリーダーエンドポイント

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 OSS (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、クラスター名を選択し、Nodes タブを選択することで、クラスターのレプリカノードエンドポイントを見つけることができます。

   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。



   Redis OSS (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:

   a. 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。

   b. 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 OSS (クラスターモードが無効) のプライマリエンドポイントは次のようになります。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
```

エンドポイントを見つける方法の詳細については、実行中のエンジンとクラスターの該当するトピックを参照してください。

- での接続エンドポイントの検索 ElastiCache
- Valkey または Redis OSS (クラスターモードが有効) クラスターのエンドポイントの検出 (コンソール) —クラスターの設定エンドポイントが必要です。
- エンドポイントの検索 (AWS CLI)
- エンドポイントの検出 (ElastiCache API)

Valkey または Redis OSS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)

必要なエンドポイントができたので、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、valkey-cli ユーティリティを使用してクラスターに接続します。最新バージョンの valkey-cli は、暗号化/認証が有効になっているクラスターを接続する SSL/TLS もサポートしています。

次の例では、Amazon Linux と Amazon Linux 2 を実行する 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 valkey-cli をインストールしてコンパイルする方法については、特定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、計画外使用のみを目的として valkey-cli ユーティリティを使用して接続をテストします。サポートされているクライアントのリストについては、「 Valkey ドキュメント」を参照してください。で を使用する AWS SDKs例については ElastiCache、「」を参照してくださいチュートリアル: Python と の開始方法 ElastiCache。

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、 を置き換えます。*primary-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey と Redis のデフォルトポートOSSは 6379 です) 。

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

   コマンドプロンプトの結果は次のようになります。

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

2. Valkey コマンドと Redis OSS 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、 を置き換えます。*configuration-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Valkey と Redis のデフォルトポートOSSは 6379 です) 。

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

> **Note**
>
> 前のコマンドでは、オプション -c は、 -ASK および -MOVED リダイレクト の後にクラスターモードを有効にします。

コマンドプロンプトの結果は次のようになります。

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

2. Valkey コマンドと Redis OSS 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になっていない場合、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、「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、valkey-cli は Valkey と 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OSS。オプションは、前のコマンドラインアクセスをダウンロードしてセットアップするセクションに示すように、valkey-cli コンパイル時に SSL/TLS BUILD_TLS=yesを有効にします。有効化AUTHはオプションです。ただし、 を有効にするには、転送中の暗号化を有効にする必要がありますAUTH。暗号化と認証の詳細については ElastiCache 、「」を参照してくださいElastiCache 転送中の暗号化 (TLS) 。

> **Note**
>
> valkey-cli `--tls`で オプションを使用して、クラスターモードが有効と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、オプションを使用してAUTHパスワード`-a`を指定できます。

次の例では、必ず を置き換えてください。*`cluster-endpoint`* また、*`port number`* クラスターのエンドポイントとポート番号。(Redis のデフォルトポートOSS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/valkey-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/valkey-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/valkey-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/valkey-cli -c -h cluster-endpoint --tls -p port number
```

クラスターに接続したら、前述の例に示すように、暗号化されていないクラスターに対して Valkey コマンドまたは Redis OSS コマンドを実行できます。

valkey-cli または Redis-cli の代替

クラスターモードが有効になっていず、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、valkey-cli または redis-cli コンパイルを経由しない場合、telnet または openssl を使用できます。次のコマンド例では、必ず を置き換えてください。*cluster-endpoint* また、*port number* クラスターのエンドポイントとポート番号。(Redis のデフォルトポートOSS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* クラスターのパスワードを使用します。

```
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
```

Valkey または Redis OSS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)

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

次の例では、valkey-cli ユーティリティを使用して、暗号化が有効になっていず、Valkey または Redis を実行しているクラスターに接続しますOSS。Valkey と使用可能なコマンドの詳細については、Valkey ウェブサイトの「Valkey コマンド」を参照してください。

valkey-cli を使用して暗号化が有効になっていない Valkey または Redis OSS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. インターネットブラウザhttps://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zipにリンクをコピーして貼り付け、 で利用可能なリリースから Redis OSSクライアントの zip ファイルをダウンロードします。 GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504

   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、Valkey ディレクトリに変更して、コマンド を実行しますc:\Valkey>valkey-cli -h *Valkey_Cluster_Endpoint* -p 6379。

   例:

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

3. Valkey または Redis OSS コマンドを実行します。

   これでクラスターに接続され、次のような Valkey コマンドまたは Redis OSS 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 valkey-cli
```

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 クラスターを削除すると、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。
> - `CreateSnapshot` 最終スナップショットを作成するには、 アクセス許可が必要です。このアクセス許可がない場合、API呼び出しは`Access Denied`例外で失敗します。

の使用 AWS Management Console

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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ElastiCache エンジンダッシュボードで、Valkey または Redis を選択しますOSS。

   エンジンで実行され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削除できるクラスターは 1 つだけです。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] で、[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。

の使用 AWS CLI

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 ElastiCache 「」トピックの「」を参照してくださいdelete-cache-cluster。

その他の ElastiCache チュートリアルと動画

以下のチュートリアルでは、Amazon ElastiCache ユーザーにとって関心のあるタスクについて説明します。

- ElastiCache 動画
- チュートリアル: Amazon ElastiCache で Amazon にアクセスするための Lambda 関数の設定 VPC

ElastiCache 動画

Amazon の基本概念と高度な ElastiCache概念を学ぶのに役立つ動画を次に示します。 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

次の動画では、Amazon を紹介します ElastiCache。

トピック

- AWS re:Invent 2020: Amazon の新機能 ElastiCache
- AWS re:Invent 2019: Amazon の新機能 ElastiCache
- AWS re:Invent 2017: Amazon の新機能 ElastiCache
- DAT204 — サービス AWS なしでSQLスケーラブルなアプリケーションを構築する (re:Invent 2015)
- DAT207 — Amazon によるアプリケーションパフォーマンスの高速化 ElastiCache （ AWS re:Invent 2013)

AWS re:Invent 2020: Amazon の新機能 ElastiCache

AWS re:Invent 2020: Amazon の新機能 ElastiCache

AWS re:Invent 2019: Amazon の新機能 ElastiCache

AWS re:Invent 2019: Amazon の新機能 ElastiCache

AWS re:Invent 2017: Amazon の新機能 ElastiCache

AWS re:Invent 2017: Amazon の新機能 ElastiCache

DAT204 — サービス AWS なしでSQLスケーラブルなアプリケーションを構築する (re:Invent 2015)

このセッションでは、NoSQL Database の利点について説明し、 AWS Amazon DynamoDB と Amazon が提供するメインの NoSQL サービスについて説明します ElastiCache。次に、Expedia と

Mapbox という 2 つの主要な顧客から、ユースケースとアーキテクチャ上の課題、および設計パターンやベストプラクティスなど、 AWS NoSQL のサービスを使用してそれらにどのように対処したかについて聞きます。このセッションでは、NoSQL とその強力な機能をより深く理解し、自信を持ってデータベースの課題に取り組む準備を整える必要があります。

DAT204 — サービス AWS なしでSQLスケーラブルなアプリケーションを構築する (re:Invent 2015)

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

このビデオでは、Amazon を使用してインメモリキャッシュシステム ElastiCache 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速化する方法について説明します。Amazon を使用してアプリケーションのレイテンシー ElastiCache を向上させ、データベースサーバーの負荷を軽減する方法を示します。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。このセッションでは、キャッシュを有効にしてメリットを得ることができるさまざまなシナリオとユースケースについて説明し、Amazon が提供する機能について説明します ElastiCache。

DAT207 - Amazon によるアプリケーションパフォーマンスの高速化 ElastiCache (re:Invent 2013)

上級者向けの動画

次の動画では、より高度な Amazon ElastiCache トピックについて説明します。

トピック

- Amazon ElastiCache のベストプラクティスを成功させるための設計 (re:Invent 2020)
- Amazon でリアルタイムアプリケーションをスーパーチャージする ElastiCache (re:Invent 2019)
- ベストプラクティス: Redis OSSクラスターを Amazon から (EC2 ElastiCache re:Invent 2019) に移行する
- Amazon ElastiCache & Amazon Aurora を使用した Fantasy Sports プラットフォームのスケーリング STP11 (re:Invent 2018)
- Amazon を使用したクラウドOSS内の信頼性とスケーラビリティに優れた Redis ElastiCache (re:Invent 2018)
- ElastiCache Deep Dive: インメモリデータストアの設計パターン (re:Invent 2018)
- DAT305 — Amazon ElastiCache Deep Dive (re:Invent 2017)
- DAT306 — Amazon ElastiCache Deep Dive (re:Invent 2016)

- DAT317— ElastiCache (Redis OSS) IFTTTを使用してイベントを予測する方法 (re:Invent 2016)
- DAT407 — Amazon ElastiCache Deep Dive (re:Invent 2015)
- SDD402 — Amazon ElastiCache Deep Dive (re:Invent 2014)
- DAT307 — Amazon ElastiCache アーキテクチャと設計パターンの詳細 (re:Invent 2013)

Amazon ElastiCache のベストプラクティスを成功させるための設計 (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 OSS、可用性、スケーラビリティ、セキュリティが最重要事項となっています。オンラインスケーリング、マルチ AZ デプロイ全体の高可用性、セキュリティ設定で Amazon を成功 ElastiCache に導くためのベストプラクティスについて説明します。

Amazon ElastiCache のベストプラクティスを成功させるための設計 (re:Invent 2020)

Amazon でリアルタイムアプリケーションをスーパーチャージする ElastiCache (re:Invent 2019)

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。フルマネージドのインメモリ Amazon 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、セキュリティ ElastiCache を実現することで、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 でリアルタイムアプリケーションをスーパーチャージする ElastiCache (re:Invent 2019:)

ベストプラクティス: Redis OSSクラスターを Amazon から (EC2 ElastiCache re:Invent 2019) に移行する

Redis OSS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 用に新しくリリースされたオンライン移行機能を使用すると ElastiCache、クラスターモードが無効になっている状態で ElastiCache、データを Amazon OSS上のセルフホスト Redis からフルマネージド Amazon EC2に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、デモを確認し、さらに重要なこととして、Amazon への移行をスムーズに行うための実践的なベストプラクティスを学びます ElastiCache。

ベストプラクティス: Redis OSSクラスターを Amazon から (EC2 ElastiCache re:Invent 2019) に移行する

Amazon ElastiCache & Amazon Aurora を使用した Fantasy Sports プラットフォームのスケーリング STP11 (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 Amit Sharma が Amazon Aurora と Amazon を使用してフラッシュトラフィック ElastiCache を処理する方法について説明します。フラッシュトラフィックは 30 秒のレスポンスウィンドウ内で 3 倍になる可能性があります。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 を使用した Fantasy Sports プラットフォームのスケーリング (STP11)

Amazon ElastiCache & Amazon Aurora を使用した Fantasy Sports プラットフォームのスケーリング STP11 (re:Invent 2018)

Amazon を使用したクラウドOSS内の信頼性とスケーラビリティに優れた Redis ElastiCache (re:Invent 2018)

このセッションでは、Redis OSS互換サービスである Amazon ElastiCache (Redis) の機能および機能強化について説明しますOSS。Redis OSS5、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の主要な機能について説明します。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 を使用したクラウドOSSでの信頼性の高いスケーラブルな Redis ElastiCache (re:Invent 2018)

ElastiCache Deep Dive: インメモリデータストアの設計パターン (re:Invent 2018)

このセッションでは、Amazon の設計とアーキテクチャについて学ぶための舞台裏を紹介します ElastiCache。Redis OSSと Memcached のサービスで一般的な設計パターンと、お客様がインメモリデータ処理にそれらをどのように使用してレイテンシーを短縮し、アプリケーションのスループットを向上させるかをご覧ください。 ElastiCache ベストプラクティス、設計パターン、アンチパターンを確認します。

ElastiCache Deep Dive: インメモリデータストアの設計パターン (re:Invent 2018)

DAT305 — Amazon ElastiCache Deep Dive (re:Invent 2017)

Amazon ElastiCacheの設計とアーキテクチャについては、舞台裏をご覧ください。Memcached と Redis OSSのサービスで一般的な設計パターンと、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ためにお客様がインメモリオペレーションにそれらをどのように使用したかをご覧ください。この動画では、 ElastiCache 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説明します。

ビデオでは次の機能を紹介しています。

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

DAT305 — Amazon ElastiCache Deep Dive (re:Invent 2017)

DAT306 — Amazon ElastiCache Deep Dive (re:Invent 2016)

Amazon ElastiCacheの設計とアーキテクチャについては、舞台裏をご覧ください。Memcached と Redis OSSのサービスで一般的な設計パターンと、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ためにお客様がインメモリオペレーションにそれらをどのように使用したかをご覧ください。このセッションでは、 ElastiCache ベストプラクティス、設計パターン、アンチパターンを確認します。

DAT306 — Amazon ElastiCache Deep Dive (re:Invent 2016)

DAT317— ElastiCache (Redis OSS) IFTTTを使用してイベントを予測する方法 (re:Invent 2016)

IFTTT は、シンプルなタスクの自動化から、自宅とのやり取りや制御方法の変革まで、ユーザーが好きなサービスをより多く活用できるようにする無料のサービスです。IFTTT は ElastiCache (Redis OSS) を使用して、トランザクション実行履歴とスケジュール予測、および Amazon S3 の ログドキュメントのインデックスを保存します。このセッションでは、Lua のスクリプティング能力と Redis のデータ型によってOSS、他の場所ではできなかったことをどのように達成できたかを説明します。

DAT317— ElastiCache (Redis OSS) IFTTTを使用してイベントを予測する方法 (re:Invent 2016)

DAT407 — Amazon ElastiCache Deep Dive (re:Invent 2015)

Amazon ElastiCacheの設計とアーキテクチャについては、舞台裏をご覧ください。Memcached および Redis OSSの一般的な設計パターンと、お客様がインメモリオペレーションにそれらをどのよう

に使用して、アプリケーションのレイテンシーとスループットを向上させたかをご覧ください。このセッションでは、Amazon に関連するベストプラクティス、設計パターン、アンチパターンを確認します ElastiCache。

DAT407 — Amazon ElastiCache Deep Dive (re:Invent 2015)

SDD402 — Amazon ElastiCache Deep Dive (re:Invent 2014)

このビデオでは、一般的なキャッシュのユースケース、Memcached エンジンと Redis OSS エンジン、ニーズに合ったエンジンを決定するのに役立つパターン、一貫したハッシュ化などについて、高速でスケーラブルなアプリケーションを構築する手段として説明します。Adobe のプリンシパルサイエンティストである Frank Wiebe は、Adobe が Amazon ElastiCache を使用してカスタマーエクスペリエンスを向上させ、ビジネスを拡大する方法について詳しく説明します。

DAT402 — Amazon ElastiCache Deep Dive (re:Invent 2014)

DAT307 — Amazon ElastiCache アーキテクチャと設計パターンの詳細 (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 OSS エンジンを比較します。このセッションでは、Amazon に関連するベストプラクティスと設計パターンも確認します ElastiCache。

DAT307 - Amazon ElastiCache アーキテクチャと設計パターンの詳細 (AWS re:Invent 2013)。

でのノードの管理 ElastiCache

ノードは、Amazon ElastiCache デプロイの最小構成要素です。これは、安全なネットワーク接続された の固定サイズのチャンクですRAM。各ノードは、クラスターの作成時または最終変更時に選択されたエンジンを実行します。各ノードには、独自のドメインネームサービス (DNS) 名とポートがあります。複数のタイプの ElastiCache ノードがサポートされており、それぞれにメモリ量と計算能力が異なります。

使用するノードサイズの詳細な説明については、「ノードサイズの選択」を参照してください。

一般的に、シャーディングのサポートにより、Valkey または Redis OSS (クラスターモードが有効) デプロイには多数の小さなノードがあります。対照的に、Valkey または Redis OSS (クラスターモードが無効) デプロイでは、クラスター内のノードが少なく、大きいノードがあります。使用するノードサイズの詳細な説明については、「ノードサイズの選択」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
- Valkey または Redis OSS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
- ノードの交換 (Valkey と Redis OSS)
- ノードの交換 (Memcached)
- 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。

- ElastiCache クラスターへのノードの追加
- ElastiCache クラスターからノードを削除する
- スケーリング ElastiCache
- での接続エンドポイントの検索 ElastiCache
- クラスター内のノードを自動的に識別する (Memcached)

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。次の手順を使用して、Amazon ElastiCache コンソール、 AWS CLI コマンド、または APIオペレーションで ElastiCache ノードステータスを表示できます。

ElastiCache ノードで可能なステータス値は、次の表のとおりです。このテーブルは、 ElastiCache ノードに対して請求されるかどうかも示します。

タイプ	請求対象	説明
`available`	請求対象	ElastiCache ノードは正常で、使用できます。
`creating`	非請求対象	ElastiCache ノードが作成されています。ノードの作成中

タイプ	請求対象	説明
		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 ノードを変更するカスタマーリクエストのため、ノードが変更されています。

タイプ	請求対象	説明
`updating`	請求対象	更新状態は、Amazon ElastiCache ノードで次のいずれかが true であることを示します。 • ElastiCache ノードは、サービス udpate の一部としてパッチ適用されています。サービスの更新の詳細については、「Amazon ElastiCache Managed Maintenance and Service Updates ヘルプ」ページを参照してください。 • VPC セキュリティグループは ElastiCache 、クラスター用に更新中です。 • 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。 • ElastiCache クラスターのログ配信設定が変更されています。 • ElastiCache ノードの削除オペレーションが保留中です。 • Valkey または Redis OSS パスワード ElastiCache を持つ は、 を使用して更新/ローテーションされていますAWS Secrets Manager。

タイプ	請求対象	説明
`rebooting cache cluster nodes`	請求対象	顧客のリクエストまたは ElastiCache ノードの再起動を必要とする Amazon ElastiCache プロセスが原因で、ノードが再起動されています。
`incompatible_parameters`	非請求対象	ノードのパラメータグループで指定されたパラメータはノードと互換性がないため、Amazon はノードを起動 ElastiCache 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。

タイプ	請求対象	説明
`incompatible_network`	非請求対象	互換性のないネットワーク状態は、Amazon ElastiCache ノードについて次のいずれかまたは複数が当てはまることを示します。 • ElastiCache ノードが起動されたサブネットに使用可能な IP アドレスはありません。 • サブネットグループに記載されている ElastiCache サブネットは、Amazon Virtual Private Cloud (Amazon) に存在しなくなりますVPC。

タイプ	請求対象	説明
restore_failed	非請求対象	復元に失敗した状態は、次のいずれかが Amazon ElastiCache ノードに当てはまることを示します。 • 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、 である前世代のノードを実行するときに発生します end-of-life。ただし、AWS が、指定されたアベイラビリティー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ーを持たない場合、現行世代のノードを置き換えることでも発生する可能性があります。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。 • 指定されたRDBスナップショットの復元に失敗しました。 • ElastiCache クラスターの AWS アカウントが停止されました。 • ノードが失敗し、復旧できませんでした。

タイプ	請求対象	説明
snapshotting	請求対象	ElastiCache は Valkey または Redis OSSノードのスナップショットを作成します。

コンソールでの ElastiCache ノードステータスの表示

コンソールで ElastiCache ノードのステータスを表示するには：

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Redis OSS 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 ElastiCache ノードのリストが表示されます。ノードごとに、ステータス値が表示されます。
3. その後、キャッシュのサービス更新タブに移動して、キャッシュに適用されるサービス更新のリストを表示できます。

を使用した ElastiCache ノードステータスの表示 AWS CLI

を使用して ElastiCache 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。例えば、次の AWS CLI コマンドは各 ElastiCache ノードを表示します。

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

を介した ElastiCache ノードステータスの表示 API

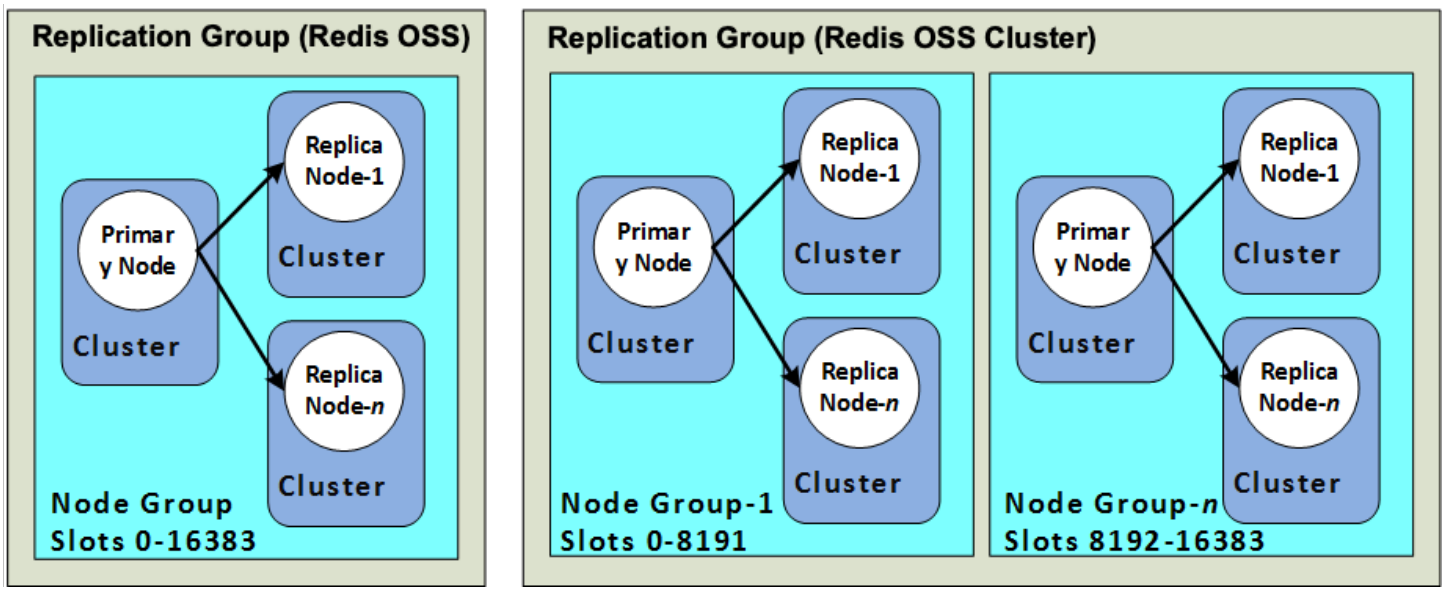
Amazon を使用して ElastiCache ノードのステータスを表示するには ElastiCache API、`ShowCacheNodeInfo`フラグ`DescribeCacheClusteroperation`を使用して を呼び出して、個々のキャッシュノードに関する情報を取得します。

Valkey または Redis OSSノードとシャード

シャード (APIおよび ではCLI、ノードグループ) はノードの階層配置であり、それぞれがクラスターでラップされます。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。Valkey または Redis OSSバージョン 3.2 以降では、クラスター内の複数のシャードがサポートされています (APIおよび ではCLI、レプリケーショングループ) 。このサポートにより、Valkey または Redis OSS (クラスターモードが有効) クラスターでデータをパーティション化できます。

次の図は、Valkey または Redis OSS (クラスターモードが無効) クラスターと Valkey または Redis OSS (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。



Valkey または Redis OSS (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 DescribeReplicationGroups (CLI: describe-replication-groups) には、メンバーノードを含むノードグループ、ノードグループ内のノードロール、およびその他の情報が一覧表示されます。

Valkey クラスターまたは Redis OSSクラスターを作成するときは、クラスター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。Valkey または Redis OSS (クラスターモードが無効) クラスターには複数のシャードはありません。シャードは、リードレプリカノードを追加 (最大 5 つ) または削除することで水平方向にスケーリングできます。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」、「Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効) 」または「Valkey または Redis のリードレプリカの削除 OSS (クラスターモードが無効) 」を参照してください。Valkey または Redis OSS (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スケールすることもできます。詳細については、「Valkey または Redis のレプリカノードのスケーリング OSS (クラスターモードが無効) 」を参照してください。

エンジンが Valkey または Redis OSSバージョン 5.0.6 以降であれば、ノードまたはシャードの制限をクラスターごとに最大 500 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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることが含まれます。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。

Valkey または Redis OSS (クラスターモードが有効) クラスターを作成したら、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、「スケーリング ElastiCache」および「ノードの交換 (Valkey と Redis OSS)」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

Valkey または Redis OSSノードへの接続

クラスター内の Valkey ノードまたは Redis OSSノードに接続しようとする前に、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。

- Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)
- Valkey または Redis OSS (クラスターモードが有効) クラスターのエンドポイントの検出 (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出 (ElastiCache API)

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

> **Note**
>
> 使用可能なコマンドの詳細については、「 コマンドウェブページ」を参照してください。

valkey-cli を使用して Valkey または Redis OSS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. を構築するにはvalkey-cli、GNUCompiler コレクション () をダウンロードしてインストールしますgcc。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、確認プロンプトyで を入力します。

   ```
   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. valkey-cli ユーティリティをダウンロードしてコンパイルします。このユーティリティは Valkey 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、次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 https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
tar xvzf valkey-7.2.6.tar.gz
cd valkey-7.2.6
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/valkey-cli -c -h mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com -p 6379
```

次のような Valkey または Redis OSS コマンドプロンプトが表示されます。

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

5. Valkey または Redis OSS 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、Valkey コマンドまたは Redis OSS コマンドを実行できます。以下は、Valkey または Redis OSSレスポンスを含むコマンドの例です。

```
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 valkey-cli
```

Secure Sockets Layer (SSL) 暗号化 (転送中有効) を持つノードまたはクラスターへの接続については、「」を参照してくださいElastiCache 転送中の暗号化 (TLS) 。

Memcached ノードへの接続

Memcached クラスター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。

- クラスターのエンドポイントの検索 (コンソール) (Memcached)
- エンドポイントの検索 (AWS CLI)
- エンドポイントの検出 (ElastiCache API)

次の例では、telnet ユーティリティを使用して Memcached を実行しているノードに接続します。

> **Note**
>
> Memcached およびその使用可能なコマンドの詳細については、Memcached のウェブサイトを参照してください。

telnet を使用してノードに接続するには

1. 任意の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. Amazon EC2インスタンスに telnet 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 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、コマンドプロンプトで y と入力します。

   ```
   sudo yum install telnet
   ```

   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: 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!
```

3. Amazon 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、ノードのエンドポイントをこの例に示すエンドポイントに置き換えます。

```
telnet mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 11211
```

以下のような出力が表示されます。

```
Trying 128.0.0.1...
Connected to mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com.
Escape character is '^]'.
>
```

4. Memcached コマンドを実行して接続をテストします。

これで、ノードに接続され、Memcached のコマンドを実行できます。次に例を示します。

```
set a 0 0 5      // Set key "a" with no expiration and 5 byte value
hello            // Set value as "hello"
STORED
get a            // Get value for key "a"
VALUE a 0 5
hello
END
get b            // Get value for key "b" results in miss
END
>
```

サポートされているノードの種類

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、「ノードサイズの選択」を参照してください。

トピック

現行世代 (Memcached)

次の表には、ネットワーク I/O クレジットメカニズムを使用して、ベースライン帯域幅を超えてバーストするインスタンスタイプのベースライン帯域幅とバースト帯域幅を示しています。

> **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 Memcached の最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m7g.large		0.937	12.5
cache.m7g.xlarge		1.876	12.5

インスタンスタイプ	サポートされている Memcached の最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m7g.2xlarge		3.75	15
cache.m7g.4xlarge		7.5	15
cache.m7g.8xlarge		15	該当なし
cache.m7g.12xlarge		22.5	該当なし
cache.m7g.16xlarge		30	該当なし
cache.m6g.large	1.5.16	0.75	10.0
cache.m6g.xlarge	1.5.16	1.25	10.0
cache.m6g.2xlarge	1.5.16	2.5	10.0
cache.m6g.4xlarge	1.5.16	5.0	10.0
cache.m6g.8xlarge	1.5.16	12	該当なし
cache.m6g.12xlarge	1.5.16	20	該当なし
cache.m6g.16xlarge	1.5.16	25	該当なし
cache.m5.large	1.5.16	0.75	10.0
cache.m5.xlarge	1.5.16	1.25	10.0
cache.m5.2xlarge	1.5.16	2.5	10.0
cache.m5.4xlarge	1.5.16	5.0	10.0
cache.m5.12xlarge	1.5.16	該当なし	該当なし
cache.m5.24xlarge	1.5.16	該当なし	該当なし
cache.m4.large	1.5.16	0.45	1.2

インスタンスタイプ	サポートされている Memcached の最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.m4.xlarge	1.5.16	0.75	2.8
cache.m4.2xlarge	1.5.16	1.0	10.0
cache.m4.4xlarge	1.5.16	2.0	10.0
cache.m4.10xlarge	1.5.16	5.0	10.0
cache.t4g.micro	1.5.16	0.064	5.0
cache.t4g.small	1.5.16	0.128	5.0
cache.t4g.medium	1.5.16	0.256	5.0
cache.t3.micro	1.5.16	0.064	5.0
cache.t3.small	1.5.16	0.128	5.0
cache.t3.medium	1.5.16	0.256	5.0
cache.t2.micro	1.5.16	0.064	1.024
cache.t2.small	1.5.16	0.128	1.024
cache.t2.medium	1.5.16	0.256	1.024

Memcached 用に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最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r7g.large		0.937	12.5
cache.r7g.xlarge		1.876	12.5
cache.r7g.2xlarge		3.75	15

インスタンスタイプ	サポートされている最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r7g.4xlarge		7.5	15
cache.r7g.8xlarge		15	該当なし
cache.r7g.12xlarge		22.5	該当なし
cache.r7g.16xlarge		30	該当なし
cache.r6g.large	1.5.16	0.75	10.0
cache.r6g.xlarge	1.5.16	1.25	10.0
cache.r6g.2xlarge	1.5.16	2.5	10.0
cache.r6g.4xlarge	1.5.16	5.0	10.0
cache.r6g.8xlarge	1.5.16	12	該当なし
cache.r6g.12xlarge	1.5.16	20	該当なし
cache.r6g.16xlarge	1.5.16	25	該当なし
cache.r5.large	1.5.16	0.75	10.0
cache.r5.xlarge	1.5.16	1.25	10.0
cache.r5.2xlarge	1.5.16	2.5	10.0
cache.r5.4xlarge	1.5.16	5.0	10.0
cache.r5.12xlarge	1.5.16	20	該当なし
cache.r5.24xlarge	1.5.16	25	該当なし
cache.r4.large	1.5.16	0.75	10.0
cache.r4.xlarge	1.5.16	1.25	10.0
cache.r4.2xlarge	1.5.16	2.5	10.0

インスタンスタイプ	サポートされている最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.r4.4xlarge	1.5.16	5.0	10.0
cache.r4.8xlarge	1.5.16	12	該当なし
cache.r4.16xlarge	1.5.16	25	該当なし

Memcached 用に最適化されたネットワーク

インスタンスタイプ	サポートされている最小バージョン	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.c7gn.large	1.6.6	6.25	30
cache.c7gn.xlarge	1.6.6	12.5	40
cache.c7gn.2xlarge	1.6.6	25	50
cache.c7gn.4xlarge	1.6.6	50	該当なし
cache.c7gn.8xlarge	1.6.6	100	該当なし
cache.c7gn.12xlarge	1.6.6	150	該当なし
cache.c7gn.16xlarge	1.6.6	200	該当なし

現行世代 (バルキーまたは Redis OSS)

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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	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
cache.m5.12xlarge	3.2.4	Y	Y	Y	12	該当なし

インスタンスタイプ	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
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
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

インスタンスタイプ	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	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
cache.r7g .16xlarge	6.2	Y	Y	Y	30	該当なし

インスタンスタイプ	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
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
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

インスタンスタイプ	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
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
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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	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	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (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	該当なし

インスタンスタイプ	サポートされる Redis の最小 OSSバージョン	拡張 I/O (Redis OSS 5.0.6+)	TLS オフロード (Redis OSS 6.2.5+)	拡張 I/O マルチプレックス (Redis OSS 7.0.4+)	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)
cache.c7gn.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

サポートされているノードタイプは、 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 の料金」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 では、汎用のバースト可能な T4g, T3-StandardT2-Standard キャッシュノードを起動できます ElastiCache。これらのノードはベースラインレベルのCPUパフォーマンスを提供し、蓄積されたクレジットを使い果たすまで、いつでもCPU使用状況をバーストできます。CPU クレジットは、フルCPUコアのパフォーマンスを 1 分間提供します。

Amazon ElastiCacheの T4g, T3、T2 ノードは標準として設定されており、平均CPU使用率が一貫してインスタンスのベースラインパフォーマンスを下回るワークロードに適しています。ベースラインを超えてバーストするには、ノードはクレジット残高に蓄積したCPU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クレジット残高が枯渇しても、ノードのパフォーマンスが急激に低下することはありません。詳細について

は、Amazon ユーザーガイドのCPU「バーストパフォーマンスインスタンスのクレジットとベースラインパフォーマンス」を参照してください。 EC2

次の表は、バーストパフォーマンスノードタイプ、1 時間あたりのCPUクレジット獲得率を示しています。また、ノードが獲得できる獲得CPUクレジットの最大数とノード vCPUs あたりの数も表示されます。さらに、ベースラインパフォーマンスレベルは、完全なコアパフォーマンス (単一の v を使用) の割合として提供されますCPU。

CPU 1 時間あたりに獲得されるクレジット	蓄積可能な最大獲得クレジット*	vCPUs	v あたりのベースラインパフォーマンスCPU	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
6	144	1	10%	0.5	低から中程度
12	288	1	20%	1.55	低から中程度

CPU 1 時間あたりに獲得されるクレジット	蓄積可能な最大獲得クレジット*	vCPUs	v あたりのベースラインパフォーマンスCPU	メモリ (GiB)	ネットワークパフォーマンス
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。

** テーブルのベースラインパフォーマンスは v ごとですCPU。複数の v を持つ一部のノードサイズCPU。これらについては、vCPU パーセンテージに の数を掛けて、ノードのベースラインCPU使用率を計算しますvCPUs。

T3 および T4g バーストパフォーマンスインスタンスでは、次のCPUクレジットメトリクスを使用できます。

> **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。

- 現行世代のすべてのノードタイプは、VPCデフォルトで Amazon に基づいて仮想プライベートクラウド (VPC) に作成されます。
- Redis OSS 追加専用ファイル (AOF) は T2 インスタンスではサポートされていません。Redis OSS設定変数 `appendonly` および `appendfsync`は、Redis OSS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 製品の機能と詳細
- Memcached の Memcached ノードタイプ固有のパラメータ
- Valkey パラメータと Redis OSSパラメータ
- 転送時の暗号化 (TLS)

ノードの再起動

一部の変更を適用するには、Redis OSSまたは Memcached クラスター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す。

トピック

- Redis OSSノードの再起動 (クラスターモードが無効のみ)
- Memcached のクラスターの再起動

Redis OSSノードの再起動 (クラスターモードが無効のみ)

Valkey または Redis OSS (クラスターモードが無効) クラスターの場合、再起動後にのみ適用されるパラメータグループのパラメータは次のとおりです。

- アクティブハッシュ化
- データベース

Redis ノードは、 ElastiCache コンソールを介してのみ更新できます。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。

> **Valkey または Redis OSS (クラスターモードが有効) パラメータの変更**
>
> Valkey または Redis OSS (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、次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. Valkey または Redis OSS (クラスターモードが有効) クラスターを削除します。「でのクラスターの削除 ElastiCache」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。

の使用 AWS Management Console

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. 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、該当する AWS リージョンを選択します。
3. 左側のナビゲーションペインで、Redis OSSを選択します。

   Redis を実行しているクラスターのリストOSS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。

Memcached のクラスターの再起動

Memcached クラスターを再起動すると、クラスターはすべてのデータをフラッシュし、エンジンを再起動します。このプロセス中はクラスターにアクセスできません。クラスターですべてのデータが

フラッシュされるため、そのクラスターもう一度利用可能になったときは、クラスターが空の状態でから開始します。

ElastiCache コンソール、、 AWS CLIまたは を使用してクラスターを再起動できます ElastiCache API。 ElastiCache コンソール、、 AWS CLI のいずれを使用する場合でも ElastiCache API、再起動を開始できるのは 1 つのクラスターのみです。複数のクラスターを再起動するには、プロセスまたはオペレーションを繰り返す必要があります。

の使用 AWS Management Console

ElastiCache コンソールを使用してクラスターを再起動できます。

クラスターを再起動するには (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. 右上隅のリストから、関心のある AWS リージョンを選択します。
3. ナビゲーションペインで、再起動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. クラスター名の左側にあるボックスを選択して、再起動するクラスターを選択します。

   [再起動] ボタンがアクティブになります。

   複数のクラスターを選択すると、[再起動] ボタンはアクティブになりません。
5. [再起動] を選択します。

   クラスターの再起動確認画面が表示されます。
6. クラスターを再起動するには、[Reboot] を選択します。クラスターの状態が、クラスターノードの再起動中に変わります。

   クラスターを再起動しない場合は、[Cancel] を選択します。

複数のクラスターを再起動するには、再起動するクラスターごとにステップ 2 ~ 5 を繰り返します。1 つのクラスターの再起動が終了するまで待たなくても、別のクラスターを再起動できます。

特定のノードを再起動するには、ノードを選択してから、[再起動] を選択します。

の使用 AWS CLI

クラスターを再起動するには (AWS CLI) 、 `reboot-cache-cluster`CLIオペレーションを使用します。

クラスターの特定のノードを再起動するには、`--cache-node-ids-to-reboot` を使用して再起動するクラスターを一覧します。次のコマンドは、my-cluster のノード 0001、0002、および 0004 を再起動します。

Linux、macOS、Unix の場合:

```
aws elasticache reboot-cache-cluster \
    --cache-cluster-id my-cluster \
    --cache-node-ids-to-reboot 0001 0002 0004
```

Windows の場合:

```
aws elasticache reboot-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --cache-node-ids-to-reboot 0001 0002 0004
```

クラスターのすべてのノードを再起動するには、`--cache-node-ids-to-reboot` パラメータを使用して、クラスターのすべてのノードの ID を選択します。詳細については、「」を参照してくださいreboot-cache-cluster。

の使用 ElastiCache API

を使用してクラスターを再起動するには ElastiCache API、 `RebootCacheCluster`アクションを使用します。

クラスターの特定のノードを再起動するには、`CacheNodeIdsToReboot` を使用して再起動するクラスターを一覧します。次のコマンドは、my-cluster のノード 0001、0002、および 0004 を再起動します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RebootCacheCluster
   &CacheClusterId=my-cluster
   &CacheNodeIdsToReboot.member.1=0001
   &CacheNodeIdsToReboot.member.2=0002
   &CacheNodeIdsToReboot.member.3=0004
   &Version=2015-02-02
```

```
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

クラスターのすべてのノードを再起動するには、CacheNodeIdsToReboot パラメータを使用して、クラスターのすべてのノードの ID を選択します。詳細については、「」を参照してくださいRebootCacheCluster。

ノードの交換 (Valkey と Redis OSS)

Amazon は ElastiCache 頻繁にフリートをアップグレードし、パッチとアップグレードをインスタンスにシームレスに適用します。ただし、基盤となるホスト ElastiCache に必須の OS 更新を適用するには、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 によって自動的に生成された代替キャッシュノードには、異なる IP アドレスがある ElastiCache 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。

次のリストは、 が Valkey または Redis OSSノードの 1 つを置き換えるように ElastiCache スケジュールするときに実行できるアクションを示しています。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 がスケジュールどおりにノードを ElastiCache 置き換えます。
- 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。
- Valkey または Redis OSS (クラスターモードが有効) の設定
  - Replace the only node in any Valkey or Redis OSS cluster - バックアップと復元を使用して Valkey または Redis OSSクラスター内のノードを置き換える手順。
  - Replace a replica node in any Valkey or Redis OSS cluster – クラスターのダウンタイムなしでレプリカ数を増減することで、任意の Valkey または Redis OSS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 Valkey or Redis OSS (cluster mode enabled) shard – スケールアウトとスケールインにより、Valkey または Redis OSS (クラスターモードが有効) クラスター内のノードを置き換えるためのクラスターダウンタイムのない動的手順。
- Valkey または Redis OSS (クラスターモードが無効) の設定

- Replace the only node in any Valkey or Redis OSS cluster - バックアップと復元を使用して Valkey または Redis OSSクラスター内のノードを置き換える手順。
- Replace a replica node in any Valkey or Redis OSS cluster – クラスターのダウンタイムなしでレプリカ数を増減することで、任意の Valkey または Redis OSS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 Valkey or Redis OSS (cluster mode disabled) cluster – レプリケーションを使用して Valkey または Redis OSS (クラスターモードが無効) クラスターのノードを置き換える手順。
- Replace a Valkey or Redis OSS (cluster mode disabled) read-replica – Valkey または Redis OSS (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。
- Replace a Valkey or Redis OSS (cluster mode disabled) primary node – Valkey または Redis OSS (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。

Valkey および Redis OSSノード置換オプション

- 何もしない – 何もしない場合、 はスケジュールどおりにノード ElastiCache を置き換えます。

自動フェイルオーバーが有効になっているクラスター以外の設定では、クラスターがオンラインのままで受信書き込みリクエストを処理する間、Valkey 7.2 以降および Redis 5.0.6 OSS 以降のクラスターが完全に置き換えられます。Redis OSS4.0.10 以下の自動フェイルオーバーが有効になっているクラスターでは、DNS更新に関連する書き込みが最大数秒短時間中断される場合があります。

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、 ElastiCache Valkey または Redis OSSを使用すると、パッチ適用、更新、その他のメンテナンス関連のノード交換時の可用性が向上します。

ElastiCache Valkey または Redis OSSクラスタークライアントで使用するように設定された Valkey または Redis クラスター設定ElastiCache では、クラスターが受信書き込みリクエストを処理する間OSS、置き換えが完了しました。

自動フェイルオーバーが有効になっているクラスター以外の設定では、クラスターがオンラインのままで受信書き込みリクエストを処理する間、Valkey 7.2 以降および Redis 5.0.6 OSS 以降のクラスターが完全に置き換えられます。Redis OSS4.0.10 以下の自動フェイルオーバーが有効になっているクラスターでは、DNS更新に関連する書き込みが最大数秒短時間中断される場合があります。

ノードがスタンドアロンの場合、Amazon は ElastiCache まず代替ノードを起動し、次に既存のノードから同期します。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、既存のノードは終了し、新しいノードが代わりに使用されます。 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の変更 – スケジュールされたメンテナンスイベントの場合、 から E メールまたは通知イベントを受け取ります ElastiCache。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代替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、 ElastiCache クラスターメンテナンスの管理 を参照してください。

- Valkey または Redis OSSクラスター内の唯一のノードを置き換える – クラスターにリードレプリカがない場合は、次の手順を使用し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、でのクラスターの削除 ElastiCache 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。

- Valkey または Redis OSSクラスターのレプリカノードを置き換える – 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。

  Valkey または Redis OSSクラスターのレプリカを置き換えるには

  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。

- Valkey または Redis OSS (クラスターモードが有効) シャードのノードを置き換える – クラスターのノードをダウンタイムなしで置き換えるには、オンライン再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。

  Valkey または Redis OSS (クラスターモードが有効) 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ディ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ディ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。

- Valkey または Redis OSS (クラスターモードが無効) クラスターのノードを置き換える – クラスターが Valkey または Redis OSS (クラスターモードが無効) クラスターで、リードレプリカがない場合は、次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、シャードなしで Valkey または Redis OSS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、ElastiCache クラスターにノードを追加するには (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、Valkey または Redis OSS (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プライマリに昇格させる 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、ElastiCache クラスターからノードを削除する 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。

- Valkey または Redis OSS (クラスターモードが無効) リードレプリカを置き換える – ノードがリードレプリカの場合は、ノードを置き換えます。

  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。

  Valkey または Redis OSS (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を減らす
     - ElastiCache クラスターからノードを削除する
  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の数を増やす
     - Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効)
  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。

- Valkey または Redis OSS (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 – ノードがプライマリノードの場合は、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。

Valkey または Redis OSS (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、Valkey または Redis OSS (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プライマリに昇格させる 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、 ElastiCache クラスターからノードを削除する を参照してください。
3. 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、Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効) 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。
5. 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。

ノードの交換 (Memcached)

Amazon ElastiCache (Memcached) は頻繁にフリートをアップグレードし、パッチとアップグレードをインスタンスにシームレスに適用します。ただし、基盤となるホストに必須の OS 更新を適用するには ElastiCache 、 (Memcached) 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頻度を手動で減らした場合には、これらのアラートを無視できます。

> **Note**
>
> Amazon によって自動的に生成された代替キャッシュノードには、異なる IP アドレスがある ElastiCache 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。

次のリストは、Memcached ノードの 1 つを置き換えるように ElastiCache スケジュールするときに実行できるアクションを示しています。

- 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置き換えます。がノード ElastiCache を自動的に新しいノードに置き換えると、新しいノードは最初は空になります。
- メンテナンスウィンドウの変更 – スケジュールされたメンテナンスイベントの場合、 から E メールまたは通知イベントを受け取りますElastiCache。この場合、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、「ElastiCache クラスターの変更」を参照してください。

  > **Note**
  >
  > メンテナンスウィンドウを移動して代替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、ElastiCache クラスターメンテナンスの管理 を参照してください。

- [手動でノードを交換する] – 次のメンテナンス時間の前にノードを交換する必要がある場合は、手動で交換します。

  ノードを手動で交換すると、キーが再配布されます。この再配布により、キャッシュミスが発生します。

  手動で Memcached ノードを交換するには

  1. 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、ElastiCache クラスターからノードを削除する を参照してください。
  2. 新しいノードをクラスターに追加します。手順については、ElastiCache クラスターへのノードの追加 を参照してください。
  3. このクラスター上で自動検出を使用していない場合は、アプリケーションで古いノードのエンドポイントのすべてのインスタンスを新しいノードのエンドポイントに置き換えます。

リザーブドノード

1 つ以上の ElastiCache ノードを予約すると、コストを削減できる場合があります。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」を参照してください。

トピック

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」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、利用可能なリザーブドノードオファリングを一覧表示して購入できます。

予約ノードの詳細については、「Amazon ElastiCache Reserved Nodes」を参照してください。

スタンダードリザーブドノードサービス

Amazon でリザーブドノードインスタンス (RI) を購入すると ElastiCache、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を取得するコミットメントを購入できます。Amazon ElastiCache 予約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の場合と同様に、新しい ElastiCache ノードインスタンスを作成します。

新しいリザーブノードインスタンスの仕様がアカウント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、リザーブドノードインスタンスに提供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。これらの標準RIsは、R5 および M5 インスタンスファミリー以降から使用できます。

> **Note**
>
> 次に説明するすべてのオファリングタイプは、1 年および 3 年単位で利用できます。

提供しているタイプ

前払い RI では、前払いを必要とせずにリザーブドElastiCache インスタンスにアクセスできます。No Upfront 予約済み ElastiCache インスタンスは、使用状況に関係なく、期間内の 1 時間ごとに割引された時間単位の料金を請求します。

部分前払い RI では、予約済み ElasticCache 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。

サイズ柔軟なリザーブドノード

すべてのリザーブドノードはサイズが柔軟です。予約済みノードを購入する場合、指定する 1 つのものはノードタイプです。例えば cache.r6g.xlarge です。ノードタイプの詳細については、「Amazon ElastiCache 料金」を参照してください。

既存の ノードがあり、これをスケールして容量を増やす必要がある場合、リザーブド ノードはスケールした ノードに自動的に適用されます。つまり、リザーブドノードは、同じノードファミリーのあらゆるサイズの使用に自動的に適用されます。同じ AWS リージョンのノードでは、サイズが柔軟なリザーブドノードを使用できます。サイズ柔軟なリザーブドノードは、そのノードファミリーでしかスケールできません。例えば、cache.r6g.xlarge の予約済みノードは cache.r6g.2xlarge に適用できますが、cache.r6g と cache.r6gd は異なるノードファミリーであるため、cache.r6g.2xlarge には適用されません。

柔軟性とは、同じノードクラスタイプ内の設定間を自由に移動できることを意味します。例えば、r6g.xlarge リザーブドノード (正規化された 8 ユニット) から同じ AWS リージョンの 2 つの

r6g.large リザーブドノード (正規化された 8 ユニット) (2*4 = 正規化された 8 ユニット) に追加料金なしで移動できます。

予約済みノードを Redis から Valkey OSSにアップグレードする

で Valkey を起動すると ElastiCache、Redis 予約OSS済みノード割引を Valkey キャッシュエンジンに適用できるようになりました。Redis から OSS Valkey にアップグレードしても、既存の契約や予約の恩恵を受けることができます。キャッシュノードファミリーとエンジン内で利点を適用できるだけでなく、より多くの増分値を受け取ることもできます。Valkey は Redis と比較して 20% の割引価格でOSS、予約ノードの柔軟性により、Redis OSS 予約ノードを使用して、実行中の Valkey ノードを 20% 以上カバーできます。

割引率を計算するために、各 ElastiCache ノードとエンジンの組み合わせには、単位単位で測定される正規化係数があります。リザーブドノードユニットは、特定のエンジン用にリザーブドノードのインスタンスファミリー内の実行中のノードに適用できます。Redis OSS 予約済みノードは、実行中の Valkey ノードをカバーするためにエンジン全体に追加で適用できます。Valkey は Redis OSSと Memcached に対して割引価格で提供されるため、特定のインスタンスタイプの単位は低く、Redis OSS 予約済みノードでより多くの Valkey ノードをカバーできます。

例えば、Redis OSS エンジン用に cache.r7g.4xlarge の予約ノードを購入し (32 ユニット)、1 つの cache.r7g.4xlarge Redis OSSノードを実行しているとします (32 ユニット)。ノードを Valkey にアップグレードすると、実行中のノードの正規化係数が 25.6 ユニットに低下し、既存の予約済みノードには、リージョンの cache.r7g ファミリー内の他の実行中の Valkey または Redis OSS ノードに対して使用する追加の 6.4 ユニットが提供されます。これを使用して、アカウントの別の cache.r7g.4xlarge Valkey ノードの 25% (25.6 ユニット)、または cache.r7g.xlarge Valkey ノードの 100% (6.4 ユニット) をカバーできます。

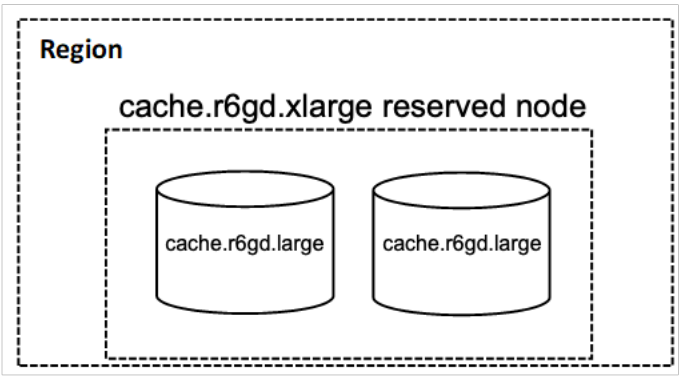
使用量と正規化された単位の比較

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

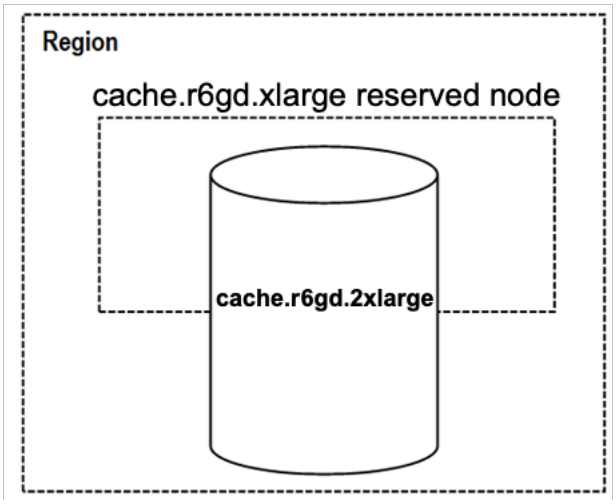
ノードサイズ	正規化された単位 (Redis OSS、Memcached)	正規化された単位 (バルキー)
micro	0.5	0.4
small	1	8.

ノードサイズ	正規化された単位 (Redis OSS、Memcached)	正規化された単位 (バルキー)
medium	2	1.6
large	4	3.2
xlarge	8	6.4
2xlarge	16	12.8
4xlarge	32	25.6
6xlarge	48	38.4
8xlarge	64	51.2
10xlarge	80	64
12xlarge	96	76.8
16xlarge	128	102.4
24xlarge	192	153.6

例えば、cache.r6gd.xlarge 予約ノードを購入し、同じ AWS リージョンのアカウントに 2 つの実行中の cache.r6gd.large 予約ノードがあるとします。この場合、料金上の利点は両方のノードに全面的に適用されます。



または、同じ AWS リージョンでアカウントで実行されている 1 つの cache.r6gd.2xlarge インスタンスがある場合、予約済みノードの使用量の 50% に請求特典が適用されます。



リザーブドノードの削除

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

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

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	中程度	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	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

提供タイプ	前払いコスト	使用料	メリット
		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。	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」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す。

次の例は、 AWS Management Console、、および を使用して AWS CLI、利用可能な予約済みノードサービスに関する料金と情報を取得する方法を示しています ElastiCache API。

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)
- 予約済みノードサービスに関する情報の取得 (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)

を使用して、利用可能なリザーブドクラスターサービスに関する料金やその他の情報を取得するには AWS Management Console、次の手順を使用します。

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. 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、[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. エンジン では、Valkey、Redis OSS、または Memcached のいずれかを選択します。
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
   - 用語
   - 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)

Valkey または Redis で利用可能な予約済みノードサービスに関する料金やその他の情報を取得するにはOSS、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では、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"
                }
            ]
}
```

Memcached で利用可能な予約済みノードサービスに関する料金やその他の情報を取得するには、コマンドプロンプトで次のコマンドを入力します。

```
 {
            "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
            "CacheNodeType": "cache.xxx.large",
            "Duration": 94608000,
            "FixedPrice": XXXX.X,
            "UsagePrice": X.X,
            "ProductDescription": "memcached",
            "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": "memcached",
            "OfferingType": "Partial Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXXX,
                    "RecurringChargeFrequency": "Hourly"
                }
```

```
            ]
        },
        {
            "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
            "CacheNodeType": "cache.xx.12xlarge",
            "Duration": 31536000,
            "FixedPrice": X.X,
            "UsagePrice": X.X,
            "ProductDescription": "memcached",
            "OfferingType": "No Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXXX,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
}
```

詳細については、 リファレンスの AWS CLI describe-reserved-cache-nodes「-offerings」を参照してください。

予約済みノードサービスに関する情報の取得 (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、 リファレンスDescribeReservedCacheNodesOfferingsの ElastiCache API「」を参照してください。

リザーブドノードの購入

次の例は、、 AWS Management Console、 AWS CLIおよび を使用して予約済みノードサービスを購入する方法を示しています ElastiCache API。

> **Important**
>
> このセクションの例に従うと、アカウントに対して元に戻す AWS ことができない料金が発生します。

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、予約済みノードオファリングをオファー ID で購入します。

リザーブドノードを購入するには

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. エンジン では、Valkey、Redis OSS、または Memcached 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
   - 用語
   - 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. [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、 リファレンスの AWS CLI purchase-reserved-cache-nodes「-offering」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。

以下のパラメータで PurchaseReservedCacheNodesOffering 操作を呼び出します。

- 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、 リファレンスPurchaseReservedCacheNodesOfferingの ElastiCache API「」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

AWS Management Console、、および を使用して AWS CLI、購入した予約済みノードに関する情報を取得できます ElastiCache API。

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)
- 予約済みノードに関する情報の取得 (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取得する方法について説明します。

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. 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取得するには、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では、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。

予約済みノードに関する情報の取得 (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、 リファレンスDescribeReservedCacheNodesの ElastiCache API「」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。前世代のノードタイプを使用する既存のクラスターがない場合、 ElastiCache はそのノードタイプを持つ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。でサポートされているノードタイプと前世代のノードタイプの詳細なリストについてはElastiCache、「」を参照してくださいサポートされているノードの種類。

Valkey または Redis OSSクラスターのノードの移行

次の手順では、 ElastiCache コンソールを使用して Valkey または Redis OSSクラスターノードタイプを移行する方法について説明します。このプロセス中、Valkey または Redis OSSクラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。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 (単一ノード) の場合、主に伝播により約 60 DNS 秒かかることがあります。
- クラスターモードが無効 (レプリカノードを使用) の場合、Valkey 7.2 以降または Redis 5.0.6 以降を実行しているクラスターでは約 OSS 1 秒が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 Valkey または Redis OSSクラスター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、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインから、Valkey クラスターまたは Redis OSS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、[すぐに適用] 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。

を使用して Valkey または Redis OSSクラスターノードタイプを変更するには AWS CLI：

modify-replication-group API 次のように 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* 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。

restore-failed-node(s) の修正または削除

次の手順では、復元に失敗したノード (複数可) を Valkey または Redis OSSクラスターで修正または削除する方法を説明します。node ElastiCache (s) が復元に失敗した状態になる方法の詳細については、「」を参照してください ElastiCache ノードステータスの表示。まず、復元に失敗した状態のノードをすべて削除し、 ElastiCache クラスター内の残りの前世代のノードを新しい世代のノードタイプに移行し、最後に必要なノード数を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、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。

2. ナビゲーションペインから、Valkey クラスターまたは Redis OSS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。

ElastiCache クラスターから復元に失敗したノード (複数可) を削除すると、新しい世代のタイプに移行できるようになりました。詳細については、上記の「Valkey または Redis OSS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するには、「」を参照してください ElastiCache クラスターへのノードの追加。

Memcached クラスターのノードの移行

ElastiCache (Memcached) を別のノードタイプに移行するには、新しいクラスターを作成する必要があります。新しいクラスターは、アプリケーションが入力できる空のクラスターから常に開始されます。

コンソールを使用して ElastiCache (Memcached) クラスターノードタイプを ElastiCache移行するには：

- 新しいノードインスタンスタイプで新しいクラスターを作成します。詳細については、「Memcached クラスター (CLI) の作成 (コンソール)」を参照してください。
- アプリケーションでは、新しいクラスターのエンドポイントにエンドポイントが更新されます。詳細については、「クラスターのエンドポイントの検索 (コンソール) (Memcached)」を参照してください。
- 古いクラスターを削除します。詳細については、「でのクラスターの削除 ElastiCache」を参照してください

でのクラスターの管理 ElastiCache

クラスターは 1 つ以上のキャッシュノードのコレクションであり、そのすべてが Valkey、Redis、OSSまたは Memcached 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。

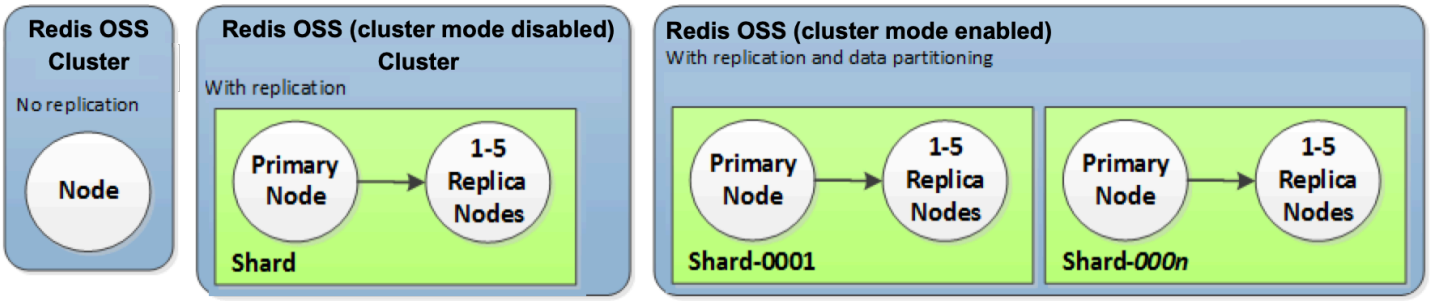
Valkey クラスターと Redis OSSクラスター

次の図は、一般的な Valkey または Redis OSSクラスターを示しています。これらのクラスターには、シャード (API/CLI: ノードグループ) 内に 1 つのノードまたは最大 6 つのノードを含めることができます。単一ノードの Valkey または Redis OSS (クラスターモードが無効) クラスターにはシャードがなく、マルチノードの Valkey または Redis OSS (クラスターモードが無効) クラスターには単一のシャードがあります。Valkey または Redis OSS (クラスターモードが有効) 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パーティション分割されます。エンジンバージョンが Valkey 7.2 以降または Redis 5.0.6 以降であれば、ノードまたはシャードの制限をクラスターごとに最大 OSS 500 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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることがあります。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。

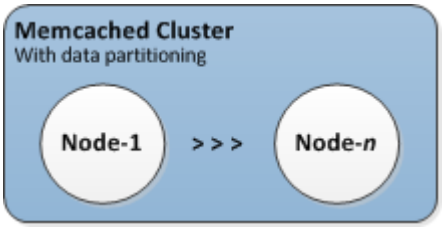
Valkey または Redis OSS シャードに複数のノードがある場合、ノードの 1 つは読み取り/書き込みプライマリノードです。シャード内の他のすべてのノードは、読み取り専用のレプリカです。

一般的な Valkey クラスターまたは Redis OSSクラスターは次のようになります。



Memcached クラスター

一般的な Memcached クラスターは次のようになります。Memcached クラスターには 1 ~ 60 個のノードがあり、データを水平にパーティション化します。



Valkey、Redis OSS、および Memcached の Elasticache オペレーション

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に特定のクラスターを指定します。クラスター識別子は、 AWS リージョン内のその顧客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されるように設計されています。Amazon VPCサービスに基づいて仮想プライベートクラウド (VPC) でクラスターを起動する場合は、 の外部からクラスターにアクセスできます AWS。詳細については、「外部から ElastiCache リソースにアクセスする AWS」を参照してください。

サポートされているバージョンのリストについては、サポートされているエンジンとバージョン、、サポートされている Redis OSSバージョンおよび を参照してくださいサポートされている ElastiCache (Memcached) バージョン。

でネットワークタイプを選択する ElastiCache

ElastiCache では、Internet Protocol バージョン 4 および 6 (IPv4 および IPv6) がサポートされているため、クラスターを次のように設定できます。

- IPv4 接続のみ、
- IPv6 接続のみ、
- IPv4 と の両方IPv6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、Valkey 7.2 以降、または Redis OSS エンジンバージョン 6.2 以降のワークロードでサポートされています。 ElastiCache 経由で にアクセスする場合、追加料金は発生しませんIPv6。

> **Note**
>
> IPV6 / デュアルスタックが利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。

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、Memcached エンジンバージョン 1.6.6 以降を使用するワークロードでサポートされています。 ElastiCache 経由で にアクセスする場合、追加料金は発生しませんIPv6。

ネットワークタイプのサブネットの設定

Amazon でクラスターを作成する場合はVPC、サブネットグループを指定する必要があります。 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ために IPv4 と IPv6 アドレスの両方が割り当てられているデュアルスタックサブネットと、 IPv6IPv6のみのサブネットが必要です。

デュアルスタックの使用

クラスターモードで ElastiCache (Redis OSS) を有効にした場合、アプリケーションの観点から、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同じです。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。

ElastiCache (Memcached) を使用してキャッシュクラスターを作成し、デュアルスタックをネットワークタイプとして選択する場合、IP 検出タイプを指定する必要があります。 IPv4 または のいずれかですIPv6。 ElastiCache はネットワークタイプと IP 検出をデフォルトとして にしますがIPv6、これは変更できます。自動検出を使用する場合、選択した IP タイプの IP アドレスのみが Memcached クライアントに返されます。詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。

すべての既存のクライアントとの下位互換性を維持するために、IP 検出が導入されます。これにより、検出プロトコルでアドバタイズする IP タイプ (IPv4または IPv6) 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クラスターモード対応のワークロードに引き続き役立ちます。これは、ダウンタイムなしで から IPv6 Discovery IP タイプIPv4への移行 (またはロールバック) を可能にするためです。

TLS 有効なデュアルスタック ElastiCache クラスター

TLS が ElastiCache クラスターで有効になっている場合、、、および などのクラスター検出関数`cluster nodes`は`cluster slots``cluster shards`、 の代わりに Valkey または Redis OSS および Memcached リターンホスト名`config get cluster`を使用しますIPs。その後、ホスト名は IPs の代わりにクラスターに接続 ElastiCache 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有効なクラスターの場合、IP Discovery パラメータは優先 IP プロトコルには影響しません。代わりに、使用される IP プロトコルは、DNS ホスト名を解決するときにクライアントがどの IP プロトコルを優先するかによって決まります。

DNS ホスト名の解決時に IP プロトコル設定を設定する方法の例については、「」を参照してくださいTLS 有効なデュアルスタック ElastiCache クラスター。

(AWS Management Console Valkey と Redis OSS) の使用

を使用してクラスターを作成するときは AWS Management Console、接続 で、ネットワークタイプまたはIPv4IPv6デュアルスタック を選択します。Valkey または Redis OSS (クラスターモードが有効) クラスターを作成し、デュアルスタックを選択する場合は、Discovery IP タイプ IPv6または を選択する必要がありますIPv4。

詳細については、Valkey または Redis OSS (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 Valkey または Redis の作成 OSS (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。

を使用してレプリケーショングループを作成するときは AWS Management Console、ネットワークタイプを またはIPv4IPv6デュアルスタック のいずれかを選択します。デュアルスタックを選択した場合は、Discovery IP タイプ IPv6または を選択する必要がありますIPv4。

詳細については、Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する または Valkey または Redis OSS (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ゼロから作成する を参照してください。

(AWS Management Console Memcached) の使用

を使用してキャッシュクラスターを作成する場合は AWS Management Console、接続 で、ネットワークタイプ またはIPv4IPv6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 タイプ IPv6または を選択する必要がありますIPv4。

詳細については、「Memcached クラスター (CLI) の作成 (コンソール)」を参照してください。

Valkey、Redis、OSSまたは Memcached CLIでの の使用

Redis OSS

OSS を使用して Valkey または Redis でキャッシュクラスターを作成する場合はCLI、 create-cache-cluster コマンドを使用して および `NetworkTypeIPDiscovery`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
```

を使用してクラスターモードが無効になっているレプリケーショングループを作成するときはCLI、create-replication-group コマンドを使用して および NetworkTypeIPDiscovery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、 を使用して IP 検出IPv4に使用する場合はCLI、 create-replication-group コマンドを使用して および NetworkTypeIPDiscovery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、 を使用して IP 検出IPv6に使用する場合はCLI、 create-replication-group コマンドを使用して および NetworkTypeIPDiscovery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
```

Memcached

を使用して Memcached でキャッシュクラスターを作成する場合はCLI、 create-cache-cluster コマンドを使用して および NetworkTypeIPDiscovery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 memcached \
    --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 memcached ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
    --ip-discovery ipv4
```

クラスター内のノードを自動的に識別する (Memcached)

Memcached エンジンを実行しているクラスターの場合、 は Auto Discovery ElastiCache をサポートします。これは、クライアントプログラムがキャッシュクラスター内のすべてのノードを自動的に識別し、これらのすべてのノードへの接続を開始および維持する機能です。

> **Note**
>
> Amazon ElastiCache Memcached で実行されているキャッシュクラスターには、Auto Discovery が追加されます。Auto Discovery は、Valkey または Redis OSS エンジンでは使用できません。

自動検出によって、アプリケーションは手動で個々のキャッシュノードに接続する必要はありません。その代わり、アプリケーションは Memcached のノードの 1 つに接続してノードのリストを取得します。そのリストからアプリケーションはクラスターの残りのノードを発見して、それらにも接続できます。アプリケーションで個々のキャッシュノードエンドポイントをハードコードする必要はありません。

クラスターでデュアルスタックネットワークタイプを使用している場合、Auto Discovery は選択したアドレスに応じて、 IPv4 または IPv6 アドレスのみを返します。詳細については、「でネットワークタイプを選択する ElastiCache」を参照してください。

クラスター内のすべてのキャッシュノードには、他のすべてのノードに関するメタデータのリストが保持されます。このメタデータは、クラスターにノードが追加または削除されるたびに更新されます。

トピック

- Memcached による Auto Discovery の利点
- 自動検出の動作
- 自動検出の使用
- マケッチドキャッシュノードへの手動接続
- Memcached クライアントライブラリへの Auto Discovery の追加
- ElastiCache 自動検出を使用するクライアント

Memcached による Auto Discovery の利点

Memcached を使用する場合、Auto Discovery には以下の利点があります。

- キャッシュクラスター内のノード数を増やすと、新しいノードは、設定エンドポイントと他のすべてのノードに自身を登録します。キャッシュクラスターからノードを削除すると、削除対象のノードが自身の登録を解除します。いずれの場合も、クラスター内の他のすべてのノードが、最新のキャッシュノードメタデータで更新されます。
- キャッシュノードの障害は自動的に検出されます。障害が発生したノードは、自動的に置き換えられます。

  > **Note**
  >
  > ノードの交換が完了するまで、そのノードは正常になりません。

- クライアントプログラムは、設定エンドポイントにのみ接続する必要があります。その後、自動検出ライブラリはクラスター内の他のすべてのノードに接続します。
- クライアントプログラムは、1 分に 1 回クラスターをポーリングします (この間隔は必要に応じて調整できます)。クラスター設定の変更がある場合 (新しいノードや削除されたノードなど)、クライアントは更新されたメタデータリストを受け取ります。その後、クライアントは必要に応じてそれらのノードに接続したり、それらのノードから切断したりします。

自動検出は、すべての ElastiCache Memcached キャッシュクラスターで有効になります。この機能を使用するためにキャッシュノードを再起動する必要はありません。

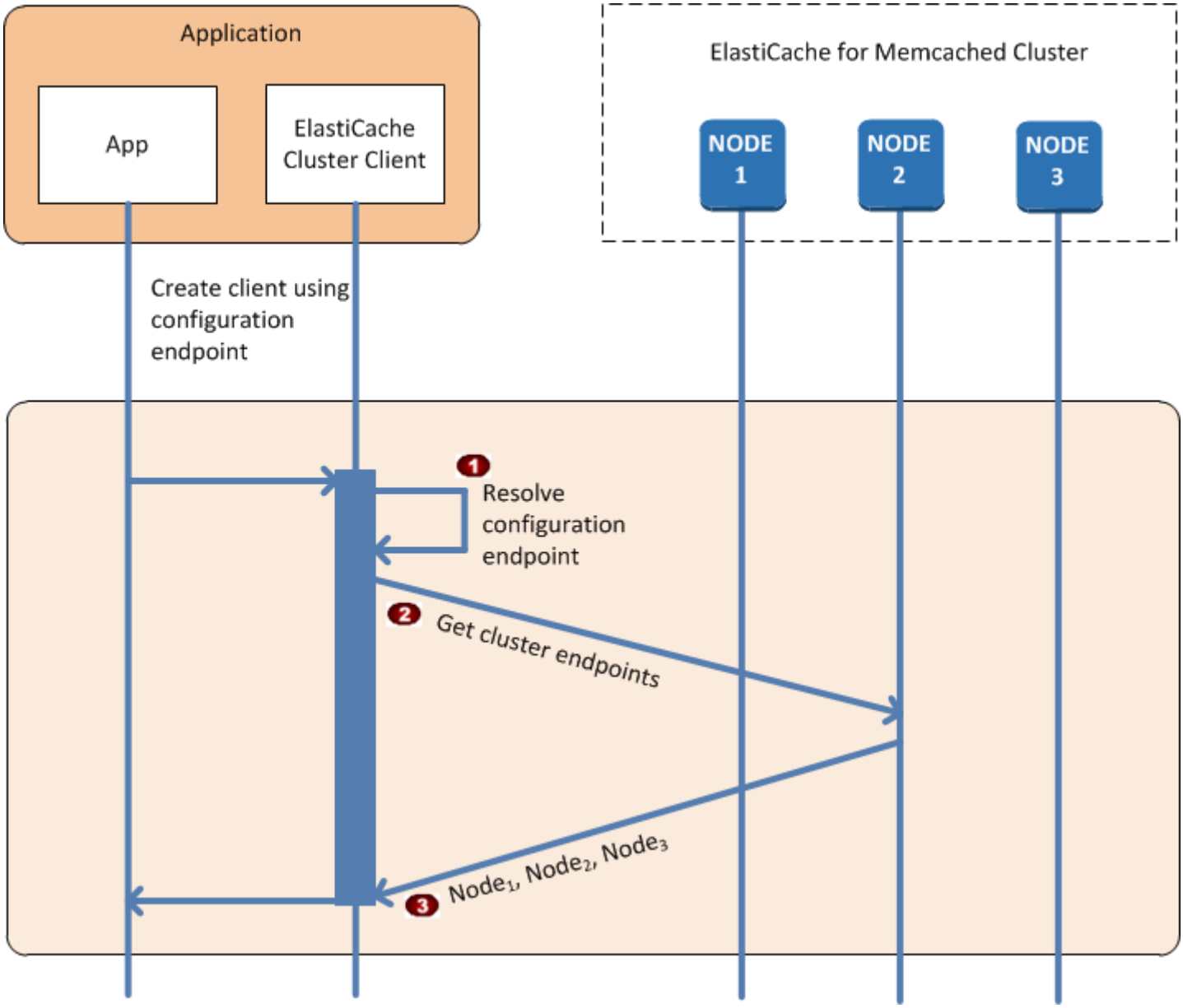
自動検出の動作

トピック

このセクションでは、クライアントアプリケーションが ElastiCache クラスタークライアントを使用してキャッシュノード接続を管理し、キャッシュ内のデータ項目を操作する方法について説明します。

キャッシュノードへの接続

アプリケーションの観点からは、クラスター設定エンドポイントへの接続は、個々のキャッシュノードに直接接続するのと変わりません。次の一連の図は、キャッシュノードに接続するプロセスを示したものです。



キャッシュノードへの接続プロセス

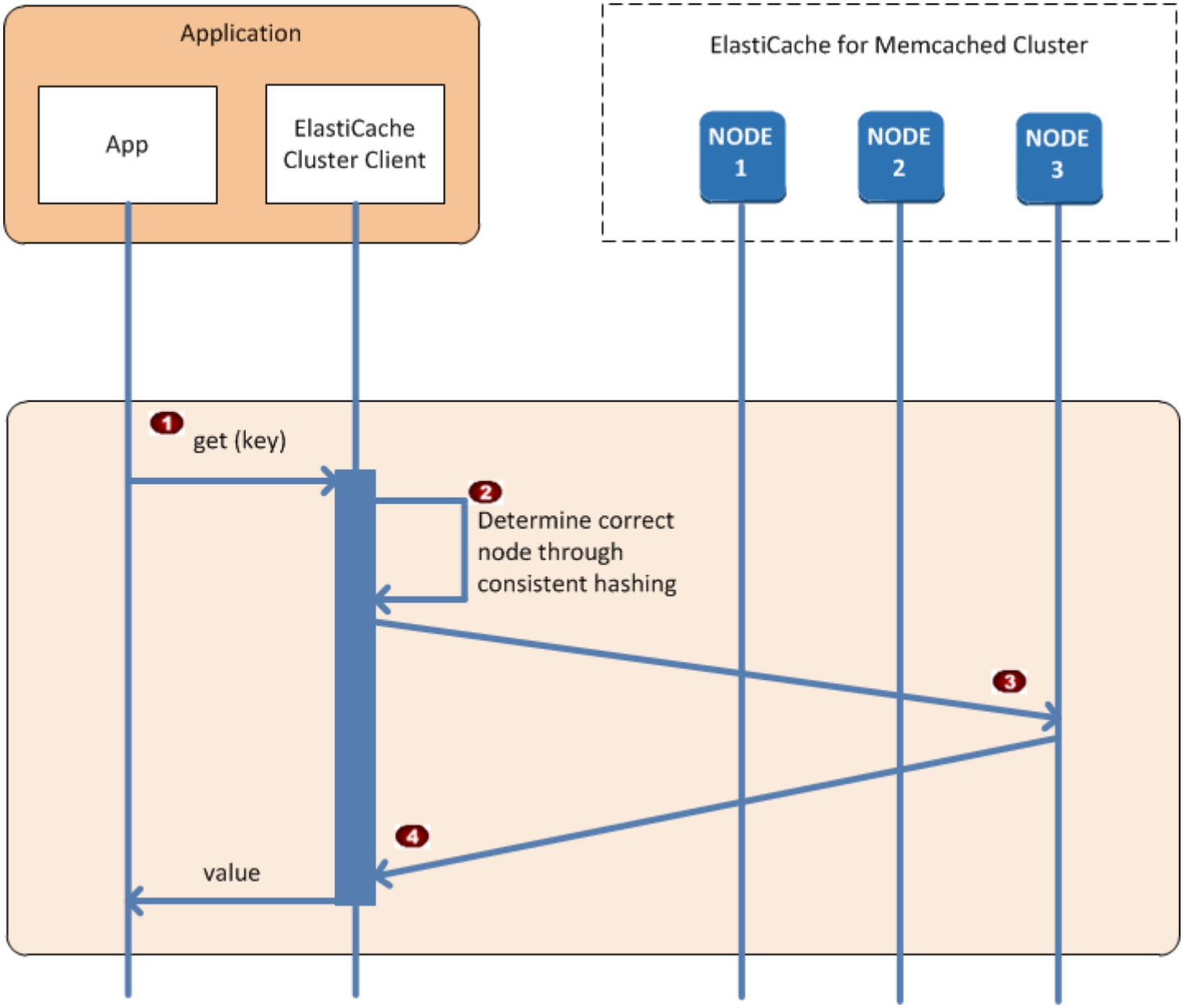
- アプリケーションは設定エンドポイントDNSの名前を解決します。設定エンドポイントはすべてのキャッシュノードのCNAMEエントリを保持するため、DNS名前はノードの 1 つに解決され、クライアントはそのノードに接続できます。
- クライアントは、他のすべてのノードの設定情報をリクエストします。各ノードにはクラスター内のすべてのノードの設定情報が保持されているため、どのノードでも必要に応じて設定情報をクライアントに渡すことができます。

- クライアントは、キャッシュノードのホスト名と IP アドレスの最新のリストを受け取ります。その後、クライアントはクラスター内の他のすべてのノードに接続できます。

> **Note**
>
> クライアントプログラムは、キャッシュノードのホスト名と IP アドレスのリストを 1 分に 1 回更新します。このポーリング間隔は、必要に応じて変更できます。

通常のクラスターオペレーション

アプリケーションがすべてのキャッシュノードに接続されると、 ElastiCache クラスタークライアントは、個々のデータ項目を保存するノードと、それらのデータ項目について後でクエリするノードを決定します。次の一連の図は、通常のクラスターオペレーションのプロセスを示しています。



通常のクラスターオペレーションのプロセス

- アプリケーションは、特定のデータ項目に対して get リクエストを発行します (キーにより識別されます)。
- クライアントは、キーに対してハッシュアルゴリズムを使用して、データ項目が格納されているキャッシュノードを調べます。
- データ項目が適切なノードからリクエストされます。
- データ項目がアプリケーションに戻ります。

その他のオペレーション

状況によっては、クラスターのノードに変更を加えることがあります。例えば、追加の需要に対応するためにノードを追加したり、需要の減少期間中にコストを節約するためにノードを削除したりできます。または、ある種類のノード障害が原因でノードを置き換えることもできます。

クラスターのエンドポイントへのメタデータ更新を必要とするクラスターを変更するときは、すべてのノードへの変更が同時に行われます。したがって、特定のノードのメタデータと、クラスター内の他のすべてのノードのメタデータの整合性がとられます。

この場合、メタデータは、クラスター内のすべてのノードで同時に更新されるため、すべてのノード間で整合性がとられます。クラスターのさまざまなノードのエンドポイントを取得するため、設定エンドポイントを必ず使用する必要があります。設定エンドポイントを使用して、「非表示」のノードからはエンドポイントデータを取得しないようにしてください。

ノードの追加

ノードがスピンアップされている間、エンドポイントはメタデータには含まれません。エンドポイントは、ノードが利用可能となった時点で、クラスターの各ノードのメタデータに追加されます。このシナリオではメタデータはすべてのノード間で整合性がとられ、新しいノードとは、それが利用可能になった後にやり取りできるようになります。ノードが利用可能になる前にはそのノードについては認識できず、新しいノードが存在しないかのようにクラスターのノードとやり取りすることになります。

ノードの削除

ノードが削除されるときは、まずエンドポイントがメタデータから削除され、ノードがクラスターから削除されます。このシナリオではメタデータはすべてのノード間で整合性がとられており、ノードが利用できない間、削除されるノードのエンドポイントがメタデータに含まれることはありません。ノードを削除している間、そのノードはメタデータでは報告されないため、アプリケーションはそのノードが存在しないかのように n-1 の残りのノードのみとやり取りします。

ノードの置換

ノードが失敗した場合、 はそのノード ElastiCache を取り出し、代替ノードをスピンアップします。この置換プロセスは数分かかります。この間、すべてのノードのメタデータには、障害のあるノードに対応するエンドポイントが表示されますが、そのノードとのやり取りの試みは失敗します。そのため、ロジックには必ず再試行ロジックを組み込んでください。

自動検出の使用

ElastiCache (Memcached) で Auto Discovery の使用を開始するには、次の手順に従います。

- 設定エンドポイントを取得する
- ElastiCache クラスタークライアントのダウンロード
- アプリケーションプログラムの変更

設定エンドポイントを取得する

クラスターに接続するには、クライアントプログラムがクラスター設定エンドポイントを認識している必要があります。トピック「クラスターのエンドポイントの検索 (コンソール) (Memcached)」を参照してください。

`--show-cache-node-info` パラメーターを指定して、`aws elasticache describe-cache-clusters` コマンドを使用することもできます。

クラスターのエンドポイント検索に使用する方法に関係なく、設定エンドポイントのアドレスには、必ず .cfg が含まれます。

Example for を使用したエンドポイントの検索 AWS CLI ElastiCache

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

Windows の場合:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

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) 。

```
{
    "CacheClusters": [
        {
            "Engine": "memcached",
            "CacheNodes": [
```

```
                {
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0001.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1e"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0002.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1a"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "mycluster",
            "PreferredAvailabilityZone": "Multiple",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-10-12T21:39:28.001Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 2,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
```

```
            "PreferredMaintenanceWindow": "sat:06:00-sat:07:00",
            "CacheNodeType": "cache.r3.large"
        }
    ]
}
```

ElastiCache クラスタークライアントのダウンロード

Auto Discovery を利用するには、クライアントプログラムでElastiCacheクラスタークライアント を使用する必要があります。 ElastiCache クラスタークライアントは Java、PHP、および で使用できます。NET には、すべてのキャッシュノードを検出して接続するために必要なロジックがすべて含まれています。

ElastiCache クラスタークライアントをダウンロードするには

1. AWS マネジメントコンソールにサインインし、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ElastiCache コンソールから、ElastiCache クラスタークライアントを選択し、ダウンロード を選択します。

ElastiCache Cluster Client for Java のソースコードは https://github.com/amazonwebservices/aws-elasticache-cluster-client-memcached-for-java で入手できます。このライブラリは、広く使用されている Spymemcached クライアントがベースとなっています。 ElastiCache クラスタークライアントは、Amazon Software License https://aws.amazon.com/asl でリリースされます。ソースコードは必要に合わせて自由に変更できます。他のオープンソース Memcached ライブラリや独自のクライアントコードにコードを組み込むこともできます。

> **Note**
>
> クラスター ElastiCache クライアントを に使用するにはPHP、まず Amazon EC2インスタンスにインストールする必要があります。詳細については、「ElastiCache Cluster Client for PHP のインストール」を参照してください。
> TLS サポートされているクライアントの場合は、PHPバージョン 7.4 以降のバイナリをダウンロードします。
> ElastiCache にクラスタークライアントを使用するにはNET、まず Amazon EC2インスタンスにクラスタークライアントをインストールする必要があります。詳細については、「のElastiCache クラスタークライアントをインストールします。NET」を参照してください。

アプリケーションプログラムの変更

自動検出を使用するようにアプリケーションプログラムを変更する準備ができました。以下のセクションでは、 ElastiCache Java、、PHPおよび 用のクラスタークライアントを使用する方法を示しますNET。

> **Important**
>
> クラスターの設定エンドポイントを指定する際は必ず、ここに示す設定エンドポイントのアドレスに「.cfg」が含まれていることを確認してください。「.cfg」なしで CNAMEまたは エンドポイントを使用しないでください。
>
> ```
> "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
> ```
>
> クラスターの設定エンドポイントを明示的に指定しない場合は、特定のノードが設定されます。

ElastiCache Cluster Client for Java の使用

以下のプログラムは、 ElastiCache クラスタークライアントを使用してクラスター設定エンドポイントに接続し、データ項目をキャッシュに追加する方法を示しています。さらに操作を行わなくても、プログラムは自動検出を使用してクラスター内のすべてのノードに接続します。

```
package com.amazon.elasticache;

import java.io.IOException;
import java.net.InetSocketAddress;

// Import the &AWS;-provided library with Auto Discovery support
import net.spy.memcached.MemcachedClient;

public class AutoDiscoveryDemo {

    public static void main(String[] args) throws IOException {

        String configEndpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
        Integer clusterPort = 11211;

        MemcachedClient client = new MemcachedClient(
                                 new InetSocketAddress(configEndpoint,
```

```
                                                        clusterPort));
        // The client will connect to the other cache nodes automatically.

        // Store a data item for an hour.
        // The client will decide which cache host will store this item.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

での ElastiCache クラスタークライアントの使用 PHP

以下のプログラムは、 ElastiCache クラスタークライアントを使用してクラスター設定エンドポイントに接続し、データ項目をキャッシュに追加する方法を示しています。さらに操作を行わなくても、プログラムは自動検出を使用してクラスター内のすべてのノードに接続します。

に ElastiCache クラスタークライアントを使用するにはPHP、まず Amazon EC2インスタンスにクラスタークライアントをインストールする必要があります。詳細については、「ElastiCache Cluster Client for PHP のインストール」を参照してください

```
<?php

 /**
  * Sample PHP code to show how to integrate with the Amazon ElastiCache
  * Auto Discovery feature.
  */

  /* Configuration endpoint to use to initialize memcached client.
   * This is only an example.  */
  $server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

  /* Port for connecting to the ElastiCache cluster.
   * This is only an example  */
  $server_port = 11211;

 /**
  * The following will initialize a Memcached client to utilize the Auto Discovery
 feature.
  *
  * By configuring the client with the Dynamic client mode with single endpoint, the
  * client will periodically use the configuration endpoint to retrieve the current
 cache
  * cluster configuration. This allows scaling the cache cluster up or down in number
 of nodes
```

```
 * without requiring any changes to the PHP application.
 *
 * By default the Memcached instances are destroyed at the end of the request.
 * To create an instance that persists between requests,
 *    use persistent_id to specify a unique ID for the instance.
 * All instances created with the same persistent_id will share the same connection.
 * See http://php.net/manual/en/memcached.construct.php for more information.
 */
 $dynamic_client = new Memcached('persistent-id');
 $dynamic_client->setOption(Memcached::OPT_CLIENT_MODE,
Memcached::DYNAMIC_CLIENT_MODE);
 $dynamic_client->addServer($server_endpoint, $server_port);

 /**
 * Store the data for 60 seconds in the cluster.
 * The client will decide which cache host will store this item.
 */
 $dynamic_client->set('key', 'value', 60);


/**
 * Configuring the client with Static client mode disables the usage of Auto Discovery
 * and the client operates as it did before the introduction of Auto Discovery.
 * The user can then add a list of server endpoints.
 */
 $static_client = new Memcached('persistent-id');
 $static_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::STATIC_CLIENT_MODE);
 $static_client->addServer($server_endpoint, $server_port);

/**
 * Store the data without expiration.
 * The client will decide which cache host will store this item.
 */
 $static_client->set('key', 'value');
 ?>
```

TLS を有効にして ElastiCache クラスタークライアントを使用する方法の例については、PHP「 と Memcached による転送中の暗号化の使用」を参照してください。

の ElastiCache クラスタークライアントの使用。NET

> **Note**
>
> ElastiCache クラスターNETクライアントは、2022 年 5 月をもって廃止されました。

。NET の クライアント ElastiCache は のオープンソースですhttps://github.com/awslabs/elasticache-cluster-config-net。

。NETアプリケーションは通常、設定ファイルから設定を取得します。サンプルアプリケーションの config ファイルを以下に示します。

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <section
            name="clusterclient"
            type="Amazon.ElastiCacheCluster.ClusterConfigSettings,
 Amazon.ElastiCacheCluster" />
    </configSections>

    <clusterclient>
        <!-- the hostname and port values are from step 1 above -->
        <endpoint hostname="mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
 port="11211" />
    </clusterclient>
</configuration>
```

以下の C# プログラムでは、 ElastiCache クラスタークライアントを使用してクラスター設定エンドポイントに接続し、データ項目をキャッシュに追加する方法を示します。さらに操作を行わなくても、プログラムは自動検出を使用してクラスター内のすべてのノードに接続します。

```
// ****************
// Sample C# code to show how to integrate with the Amazon ElastiCcache Auto Discovery
 feature.

using System;

using Amazon.ElastiCacheCluster;
```

```
using Enyim.Caching;
using Enyim.Caching.Memcached;

public class DotNetAutoDiscoveryDemo  {

    public static void Main(String[] args)  {

        // instantiate a new client.
        ElastiCacheClusterConfig config = new ElastiCacheClusterConfig();
        MemcachedClient memClient = new MemcachedClient(config);

        // Store the data for 3600 seconds (1hour) in the cluster.
        // The client will decide which cache host will store this item.
        memClient.Store(StoreMode.Set, 3600, "This is the data value.");

    }  // end Main

}  // end class DotNetAutoDiscoverDemo
```

マケッチドキャッシュノードへの手動接続

クライアントプログラムで Auto Discovery が使用されていない場合は、Memcached の各キャッシュノードに手動で接続できます。これは、Memcached クライアントのデフォルトの動作です。

キャッシュノードのホスト名とポート番号のリストは、AWS マネジメントコンソールから取得できます。`--show-cache-node-info` パラメータで コマンドを使用 AWS CLI `aws elasticache describe-cache-clusters`することもできます。

Example

以下の Java コードスニペットは、4 ノードキャッシュクラスター内のすべてのノードに接続する方法を示しています。

```
...

ArrayList<String> cacheNodes = new ArrayList<String>(
 Arrays.asList(
     "mycachecluster.fnjyzo.0001.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0002.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0003.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0004.use1.cache.amazonaws.com:11211"));

MemcachedClient cache = new MemcachedClient(AddrUtil.getAddresses(cacheNodes));

...
```

> **Important**
>
> ノードを追加または削除することでキャッシュクラスターをスケールアップまたはスケールダウンする場合、クライアントコード内のノードのリストを更新する必要があります。

Memcached クライアントライブラリへの Auto Discovery の追加

Auto Discovery の設定情報は、各 Memcached キャッシュクラスターノードに冗長的に保存されます。クライアントアプリケーションは、任意のキャッシュノードのクエリを実行し、クラスター内のすべてのノードの設定情報を取得できます。

アプリケーションがこれを行う方法は、キャッシュエンジンバージョンによって異なります。

- キャッシュエンジンバージョンが 1.4.14 以上の場合、`config` コマンドを使用します。
- キャッシュエンジンバージョンが 1.4.14 未満の場合、`get AmazonElastiCache:cluster` コマンドを使用します。

これらの 2 つのコマンドの出力は同じです。以下の「[Output Format] (出力形式)」セクションで説明します。

キャッシュエンジンバージョン 1.4.14 以上

キャッシュエンジンバージョン 1.4.14 以上の場合、`config` コマンドを使用します。このコマンドは、 によって Memcached ASCIIプロトコルとバイナリプロトコルに追加され ElastiCache、ElastiCache クラスタークライアントに実装されています。別のクライアントライブラリで自動検出を使用する場合、`config` コマンドをサポートするためにそのライブラリを拡張する必要があります。

> **Note**
>
> 次のドキュメントはASCIIプロトコルに関連していますが、 `config` コマンドは ASCIIと バイナリの両方をサポートしています。バイナリプロトコルを使用して Auto Discovery サポートを追加する場合は、 ElastiCache クラスタークライアント のソースコードを参照してください。

[Syntax] (構文)

```
config [sub-command] [key]
```

オプション

名前	説明	必須
sub-command	キャッシュノードの操作に使用されるサブコマンド。自動検出の場合、このサブコマンドは get です。	可能
key	クラスター設定が格納されたキー。自動検出の場合、このキーの名前は cluster です。	可能

クラスターの設定情報を取得するには、以下のコマンドを使用します。

```
config get cluster
```

キャッシュエンジンバージョン 1.4.14 未満

クラスターの設定情報を取得するには、以下のコマンドを使用します。

```
get AmazonElastiCache:cluster
```

> **Note**
>
> クラスター設定情報が存在する場所であるため、AmazonElastiCache「:cluster」キーを改ざんしないでください。このキーを上書きすると、クライアントが短時間 (15 秒以下) 誤って設定されてから、設定情報が ElastiCache 自動的かつ正しく更新される可能性があります。

[Output Format] (出力形式)

config get cluster を使用するか get AmazonElastiCache:cluster を使用するかにかかわらず、応答は 2 行で構成されます。

- 設定情報のバージョン番号。キャッシュクラスターにノードが追加または削除されるたび、バージョン番号は 1 ずつ増加します。
- キャッシュノードのリスト。リスト内の各ノードは、hostname|ip-address|port グループによって表され、各ノードはスペースで区切られます。

各行の末尾には、キャリッジリターンと改行文字 (CR + LF) が表示されます。データ行には最後に改行文字 (LF) が含まれ、ここに CR + LF が追加されます。バージョン設定行は、CR なしの LF で終了します。

3 つのノードを含むキャッシュクラスターは、次のように表されます。

```
configversion\n
hostname|ip-address|port hostname|ip-address|port hostname|ip-address|port\n\r\n
```

各ノードには、 CNAMEとプライベート IP アドレスの両方が表示されます。CNAME は常に存在します。プライベート IP アドレスが使用できない場合は表示されませんが、パイプ文字 |「」は引き続き印刷されます。

Example

設定情報のクエリを実行した場合に返されるペイロードの例を次に示します。

```
CONFIG cluster 0 136\r\n
12\n
myCluster.pc4ldq.0001.use1.cache.amazonaws.com|10.82.235.120|11211
 myCluster.pc4ldq.0002.use1.cache.amazonaws.com|10.80.249.27|11211\n\r\n
END\r\n
```

> **Note**
>
> - 2 行目は、設定情報がこれまで 12 回変更されたことを示しています。
> - 3 行目のノードのリストでは、ホスト名がアルファベット順に並んでいます。この順序は、現在クライアントアプリケーションで何を使用しているかにより異なる場合があります。

ElastiCache 自動検出を使用するクライアント

クラスタークライアントプログラムは、Memcached エンジンを実行しているすべてのキャッシュクラスターノードを自動的に識別して接続できます。

このセクションでは、 と のインストールと設定 ElastiCache PHPについて説明します。NET クライアントは、自動検出で使用します。

トピック

- クラスタークライアントのインストールとコンパイル
- ElastiCache クライアントの設定

クラスタークライアントのインストールとコンパイル

このセクションでは、 PHP と のインストール、設定、コンパイルについて説明します。NET Amazon ElastiCache 自動検出クラスタークライアント。

トピック

- の ElastiCache クラスタークライアントをインストールします。NET
- ElastiCache Cluster Client for PHP のインストール
- の ElastiCache クラスタークライアントのソースコードのコンパイル PHP

の ElastiCache クラスタークライアントをインストールします。NET

は にあります ElastiCache 。NET でクライアントコードをオープンソースとしてクラスター化しますhttps://github.com/awslabs/elasticache-cluster-config-net。

このセクションでは、Amazon EC2インスタンスのクラスタークライアントの .NET コンポーネント ElastiCacheをインストール、更新、削除する方法について説明します。自動検出の詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。サンプル 。NETクライアントを使用するコードについては、「」を参照してくださいの ElastiCache クラスタークライアントの使用。NET。

トピック

- のインストール。NET
- の ElastiCache .NET クラスタークライアントをダウンロードする ElastiCache
- で AWS アセンブリをインストールする NuGet

のインストール。NET

が必要です。NET 3.5 以降がインストールされ、 AWS SDKに NETを使用します ElastiCache。をお持ちでない場合。NET 3.5 以降では、http://www.microsoft.com/net から最新バージョンをダウンロードしてインストールできます。

の ElastiCache .NET クラスタークライアントをダウンロードする ElastiCache

ElastiCache .NET クラスタークライアントをダウンロードするには

1. にサインイン AWS Management Console し、 https://console.aws.amazon.com/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、ElastiCache クラスタークライアント をクリックします。
3. ElastiCache Memcached クラスタークライアントのダウンロードリストで、.NET を選択し、ダウンロードをクリックします。

で AWS アセンブリをインストールする NuGet

NuGet は、 NETプラットフォームのパッケージ管理システムです。NuGet は、アセンブリの依存関係を認識し、必要なすべてのファイルを自動的にインストールします。 NuGet インストールされたアセンブリは、 などの一元的な場所ではなく、 ソリューションに保存されるためProgram Files、互換性の問題を発生させることなく、アプリケーションに固有のバージョンをインストールできます。

のインストール NuGet

NuGet は のインストールギャラリーからインストールできますMSDN。https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c を参照してください。Visual Studio 2010 以降を使用している場合、 は自動的にインストール NuGet されます。

Solution Explorer または Package Manager コンソール NuGet から使用できます。

Solution Explorer NuGet からの の使用

Visual Studio 2010 で Solution Explorer NuGet から を使用するには

1. [Tools] メニューから、[Library Package Manager] を選択します。
2. [Package Manager Console] をクリックします。

Visual Studio 2012 または Visual Studio 2013 で Solution Explorer NuGet から を使用するには

1. Tools メニューから、NuGet Package Manager を選択します。
2. [Package Manager Console] をクリックします。

コマンドラインから、次のように Install-Package を使用してアセンブリをインストールできます。

```
Install-Package Amazon.ElastiCacheCluster
```

や AWS拡張アセンブリなど NuGet、 を通じて利用可能なすべてのパッケージの AWS SDKページを確認するには、http://www.nuget.org NuGet のウェブサイトを参照してください。各パッケージのページには、 コンソールを使用してパッケージをインストールするためのコマンドラインのサンプルと、 で利用可能なパッケージの以前のバージョンのリストが含まれていますNuGet。

Package Manager Console のコマンドの詳細については、http://nuget.codeplex.com/wikipage?title=Package%20Manager%20Console%20Command%20Reference%20%28v1.3%29 を参照してください。

ElastiCache Cluster Client for PHP のインストール

このセクションでは、Amazon EC2 インスタンスで ElastiCache Cluster Client の PHP コンポーネントをインストール、更新、および削除する方法について説明します。自動検出の詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。クライアントを使用するサンプル PHP コードについては、「での ElastiCache クラスタークライアントの使用 PHP」を参照してください。

トピック

- インストールパッケージのダウンロード
- 既に php-memcached 拡張機能をインストールしているユーザーの場合
- 新規ユーザーのインストール手順
- PHP クラスタークライアントの削除

インストールパッケージのダウンロード

に ElastiCache クラスタークライアントの正しいバージョンを使用するようにするには PHP、Amazon EC2インスタンスにインストールPHPされている のバージョンを確認する必要があります。また、Amazon EC2インスタンスが Linux の 64 ビットバージョンまたは 32 ビットバージョンを実行しているかどうかを知る必要があります。

Amazon EC2インスタンスにインストールされているPHPバージョンを確認するには

- コマンドプロンプトで、次のコマンドを入力します。

```
php -v
```

この例のように、PHPバージョンが出力に表示されます。

```
PHP 5.4.10 (cli) (built: Jan 11 2013 14:48:57)
Copyright (c) 1997-2012 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2012 Zend Technologies
```

> **Note**
>
> PHP と Memcached のバージョンに互換性がない場合、次のようなエラーメッセージが表示されます。

```
PHP Warning: PHP Startup: memcached: Unable to initialize module
Module compiled with module API=20100525
PHP compiled with module API=20131226
These options need to match
in Unknown on line 0
```

この場合は、ソースコードからモジュールをコンパイルする必要があります。詳細については、「の ElastiCache クラスタークライアントのソースコードのコンパイル PHP」を参照してください。

Amazon EC2AMIアーキテクチャ (64 ビットまたは 32 ビット) を確認するには

1. にサインイン AWS Management Console し、 で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. インスタンスリストで、Amazon EC2インスタンスをクリックします。
3. 説明タブで、AMI： フィールドを探します。64 ビットのインスタンスでは、説明に x86_64 が含まれています。32 ビットのインスタンスの場合は、このフィールドで i386 または i686 を探します。

これで、 ElastiCache クラスタークライアントをダウンロードする準備が整いました。

の ElastiCache クラスタークライアントをダウンロードするには PHP

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. ElastiCache コンソールから、ElastiCache クラスタークライアント を選択します。
3. ElastiCache Memcached クラスタークライアントのダウンロードリストから、PHPバージョンとAMIアーキテクチャに一致する ElastiCache クラスタークライアントを選択し、ダウンロードボタンを選択します。

既に php-memcached 拡張機能をインストールしているユーザーの場合

php-memcached のインストールを更新するには

1. 「PHP クラスタークライアントの削除」トピックで説明しているように、PHP 用の Memcached 拡張機能の以前のインストールを削除します。
2. 前に「新規ユーザーのインストール手順」で説明したように、新しい ElastiCache php-memcached 拡張機能をインストールします。

新規ユーザーのインストール手順

トピック

- 新規ユーザー向けの 7.x PHP のインストール
- 新規ユーザー用の 5.x PHP のインストール

新規ユーザー向けの 7.x PHP のインストール

トピック

- Ubuntu サーバー 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP7 をインストールするには
- Amazon Linux 201609 に PHP 7 をインストールするには AMI
- SUSE Linux に 7 PHP をインストールするには AMI

Ubuntu サーバー 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP7 をインストールするには

1. から新しいインスタンスを起動しますAMI。
2. 以下のコマンドを実行します。

```
sudo apt-get update
sudo apt-get install gcc g++
```

3. 7 PHP をインストールします。

```
sudo yum install php70
```

4. Amazon ElastiCache クラスタークライアントをダウンロードします。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. latest-64bit を展開します。

```
tar -zxvf latest-64bit
```

6. root アクセス権限を使用して、抽出されたアーティファクトファイル amazon-elasticache-cluster-client.so を /usr/lib/php/20151012 にコピーします。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib/php/20151012
```

7. /etc/php/7.0/cli/php.ini ファイルに extension=amazon-elasticache-cluster-client.so 行を挿入します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php/7.0/cli/php.ini
```

8. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

Amazon Linux 201609 に PHP 7 をインストールするには AMI

1. から新しいインスタンスを起動しますAMI。
2. 次のコマンドを実行します。

```
sudo yum install gcc-c++
```

3. 7 PHP をインストールします。

```
sudo yum install php70
```

4. Amazon ElastiCache クラスタークライアントをダウンロードします。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. latest-64bit を展開します。

```
tar -zxvf latest-64bit
```

6. root アクセス権限を使用して、抽出されたアーティファクトファイル amazon-elasticache-cluster-client.so を /usr/lib64/php/7.0/modules/ にコピーします。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.0/modules/
```

7. 50-memcached.ini ファイルを作成します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php-7.0.d/50-memcached.ini
```

8. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

SUSE Linux に 7 PHP をインストールするには AMI

1. から新しいインスタンスを起動しますAMI。
2. 次のコマンドを実行します。

```
sudo zypper install gcc
```

3. 7 PHP をインストールします。

```
sudo yum install php70
```

4. Amazon ElastiCache クラスタークライアントをダウンロードします。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. latest-64bit を展開します。

```
tar -zxvf latest-64bit
```

6. root アクセス権限を使用して、抽出されたアーティファクトファイル amazon-elasticache-cluster-client.so を /usr/lib64/php7/extensions/ にコピーします。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php7/extensions/
```

7. /etc/php7/cli/php.ini ファイルに extension=amazon-elasticache-cluster-client.so 行を挿入します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php7/cli/php.ini
```

8. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

新規ユーザー用の 5.x PHP のインストール

トピック

- Amazon Linux 2014.03 (64 ビットおよび 32 ビット) に AMI PHP5 をインストールするには
- Red Hat Enterprise Linux 7.0 AMI (64 ビットおよび 32 ビット) に PHP5 をインストールするには
- Ubuntu サーバー 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP5 をインストールするには
- Linux SUSEエンタープライズサーバー 11 AMI (64 ビットまたは 32 ビット) 用の PHP5 をインストールするには
- 他の Linux ディストリビューション

Amazon Linux 2014.03 (64 ビットおよび 32 ビット) に AMI PHP5 をインストールするには

1. Amazon Linux インスタンス (64 ビットまたは 32 ビット) を起動し、ログインします。
2. PHP 依存関係のインストール：

```
sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2インスタンスとPHPバージョンに適したphp-memcachedパッケージをダウンロードします。詳細については、「インストールパッケージのダウンロード」を参照してください。

4. php-memcached をインストールします。は、インストールパッケージのダウンロードパスURIである必要があります。

```
sudo pecl install <package download path>
```

PHP 5.4、64 ビット Linux のインストールコマンドの例を次に示します。このサンプルでは、*X.Y.Z* 実際のバージョン番号：

```
sudo pecl install /home/AmazonElastiCacheClusterClient-X.Y.Z-PHP54-64bit.tgz
```

> **Note**
>
> インストールアーティファクトの最新バージョンを使用してください。

5. root/sudo アクセス許可を使用して、 /etc/php.d ディレクトリmemcached.iniに という名前の新しいファイルを追加し、ファイルに「extension=amazon-elasticache-cluster-client.so」を挿入します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php.d/memcached.ini
```

6. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

Red Hat Enterprise Linux 7.0 AMI (64 ビットおよび 32 ビット) に PHP5 をインストールするには

1. Red Hat Enterprise Linux インスタンス (64 ビットまたは 32 ビット) を起動し、ログインします。
2. PHP 依存関係のインストール：

```
sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2インスタンスとPHPバージョンに適したphp-memcachedパッケージをダウンロードします。詳細については、「インストールパッケージのダウンロード」を参照してください。

4. php-memcached をインストールします。は、インストールパッケージのダウンロードパスURIである必要があります。

```
sudo pecl install <package download path>
```

5. root/sudo アクセス許可を使用して、memcached.ini という名前の新しいファイルを /etc/php.d ディレクトリに追加し、このファイルに「extension=amazon-elasticache-cluster-client.so」を挿入します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/php.d/memcached.ini
```

6. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

Ubuntu サーバー 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP5 をインストールするには

1. Ubuntu Linux インスタンス (64 ビットまたは 32 ビット) を起動し、ログインします。
2. PHP 依存関係のインストール：

```
sudo apt-get update
sudo apt-get install gcc g++ php5 php-pear
```

3. Amazon EC2インスタンスとPHPバージョンに適したphp-memcachedパッケージをダウンロードします。詳細については、「インストールパッケージのダウンロード」を参照してください。
4. php-memcached をインストールします。は、インストールパッケージのダウンロードパスURIである必要があります。

```
sudo pecl install <package download path>
```

> **Note**
>
> このインストール手順では、ビルドアーティファクト amazon-elasticache-cluster-client.so が /usr/lib/php5/20121212* ディレクトリにインストール

されます。次のステップで必要になるため、ビルドアーティファクトの絶対パスを確認してください。

前のコマンドが機能しない場合は、ダウンロードした*.tgzファイルamazon-elasticache-cluster-client.soからPHPクライアントアーティファクトを手動で抽出し、/usr/lib/php5/20121212*ディレクトリにコピーする必要があります。

```
tar -xvf <package download path>
cp amazon-elasticache-cluster-client.so /usr/lib/php5/20121212/
```

5. root/sudo アクセス許可を使用して、 /etc/php5/cli/conf.d ディレクトリmemcached.iniに という名前の新しいファイルを追加し、ファイルに「extension=<absolute path to amazon-elasticache-cluster-client.so>」を挿入します。

```
echo "extension=<absolute path to amazon-elasticache-cluster-client.so>" | sudo tee
 --append /etc/php5/cli/conf.d/memcached.ini
```

6. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

Linux SUSEエンタープライズサーバー 11 AMI (64 ビットまたは 32 ビット) 用の PHP5 をインストールするには

1. SUSE Linux インスタンス (64 ビットまたは 32 ビット) を起動し、ログインします。
2. PHP 依存関係のインストール：

```
sudo zypper install gcc php53-devel
```

3. Amazon EC2インスタンスとPHPバージョンに適したphp-memcachedパッケージをダウンロードします。詳細については、「インストールパッケージのダウンロード」を参照してください。
4. php-memcached をインストールします。は、インストールパッケージのダウンロードパスURIである必要があります。

```
sudo pecl install <package download path>
```

5. root/sudo アクセス許可を使用して、memcached.ini という名前の新しいファイルを /etc/php5/conf.d ディレクトリに追加し、このファイルに「**extension=amazon-elasticache-cluster-client.so**」を挿入します。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/php5/conf.d/memcached.ini
```

6. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

> **Note**
>
> ステップ 5 が以前のプラットフォームで機能しない場合、amazon-elasticache-cluster-client.so のインストールパスを確認してください。さらに、extension でバイナリの完全なパスを指定します。さらに、使用中PHPの がサポートされているバージョンであることを確認します。バージョン 5.3 ~ 5.5 がサポートされています。

他の Linux ディストリビューション

一部のシステムでは、特に CentOS7 と Red Hat Enterprise Linux (RHEL) 7.1 libsasl2.so.3が を置き換えましたlibsasl2.so.2。これらのシステムでは、 ElastiCache クラスタークライアントをロードすると、 の検索とロードが試行され、失敗しますlibsasl2.so.2。この問題を解決するには、クライアントが libsasl2.so.2 をロードしようとしたときに libsasl2.so.3 にリダイレクトされるように、libsasl2.so.3 へのシンボリックリンクを作成します。次のコードでは、このシンボリックリンクが作成されます。

```
cd /usr/lib64
sudo ln libsasl2.so.3 libsasl2.so.2
```

PHP クラスタークライアントの削除

トピック

- 以前のバージョンの 7 PHP の削除
- 以前のバージョンの 5 PHP を削除する

以前のバージョンの 7 PHP の削除

以前のバージョンの 7 PHP を削除するには

1. インストール手順で前述したように、適切な PHP lib ディレクトリからamazon-elasticache-cluster-client.soファイルを削除します。「既に php-memcached 拡張機能をインストールしているユーザーの場合」でインストールのセクションを参照してください。
2. php.ini ファイルから extension=amazon-elasticache-cluster-client.so 行を削除します。
3. Apache サーバーを起動または再起動します。

```
sudo /etc/init.d/httpd start
```

以前のバージョンの 5 PHP を削除する

以前のバージョンの 5 PHP を削除するには

1. php-memcached 拡張機能を削除します。

```
sudo pecl uninstall __uri/AmazonElastiCacheClusterClient
```

2. 前のインストールの手順に従って適切なディレクトリに追加した memcached.ini ファイルを削除します。

の ElastiCache クラスタークライアントのソースコードのコンパイル PHP

このセクションでは、 の ElastiCache クラスタークライアントのソースコードを取得してコンパイルする方法について説明しますPHP。

プル GitHub してコンパイルする必要があるパッケージは aws-elasticache-cluster-client-libmemcached と aws-elasticache-cluster-client-memcached-for-php の 2 つです。

トピック

- libmemcached ライブラリのコンパイル
- の ElastiCache Memcached 自動検出クライアントのコンパイル PHP

libmemcached ライブラリのコンパイル

aws-elasticache-cluster-client-libmemcached ライブラリをコンパイルするには

1. Amazon EC2インスタンスを起動します。
2. ライブラリの依存関係をインストールします。
   - Amazon Linux 201509 の場合 AMI

     ```
     sudo yum install gcc gcc-c++ autoconf libevent-devel
     ```

   - Ubuntu 14.04 の場合 AMI

     ```
     sudo apt-get update
     sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev
     ```

3. リポジトリをプルし、コードをコンパイルします。

   ```
   Download and install  https://github.com/awslabs/aws-elasticache-cluster-client-
   libmemcached/archive/v1.0.18.tar.gz
   ```

の ElastiCache Memcached 自動検出クライアントのコンパイル PHP

以下のセクションでは、 ElastiCache Memcached Auto Discovery Client をコンパイルする方法について説明します。

トピック

- 7 の ElastiCache Memcached PHP クライアントをコンパイルする
- 5 の ElastiCache Memcached PHP クライアントをコンパイルする

7 の ElastiCache Memcached PHP クライアントをコンパイルする

code ディレクトリで以下の一連のコマンドを実行します。

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php.git
cd aws-elasticache-cluster-client-memcached-for-php
git checkout php7
sudo yum install php70-devel
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-
memcached-sasl
make
make install
```

> **Note**
>
> libmemcached ライブラリをPHPバイナリに静的にリンクして、さまざまな Linux プラットフォームに移植できます。そのためには、make の前にコマンドを実行します。
>
> ```
> sed -i "s#-lmemcached#<libmemcached-install-directory>/lib/libmemcached.a -
> lcrypt -lpthread -lm -lstdc++ -lsasl2#" Makefile
> ```

5 の ElastiCache Memcached PHP クライアントをコンパイルする

aws-elasticache-cluster-client-memcached-for-php/ フォルダーで、以下のコマンドを実行して aws-elasticache-cluster-client-memcached-for-php をコンパイルします。

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php.git
cd aws-elasticache-cluster-client-memcached-for-php
sudo yum install zlib-devel
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory>
make
```

```
make install
```

ElastiCache クライアントの設定

ElastiCache クラスターは、Valkey、Redis OSS、および Memcached とプロトコルに準拠しています。既存の環境で現在使用しているコード、アプリケーション、および最も一般的なツールは、サービスとシームレスに連携します。

このセクションでは、 のキャッシュノードに接続するための具体的な考慮事項について説明します ElastiCache。

トピック

制限されるコマンド

マネージドサービスエクスペリエンスを提供するために、 は、高度な権限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 ElastiCache を制限します。Redis を実行するキャッシュクラスターの場合、次のコマンドは使用できません。

- `bgrewriteaof`
- `bgsave`
- `config`
- `debug`
- `migrate`
- `replicaof`
- `save`
- `slaveof`
- `shutdown`
- `sync`

ノードのエンドポイントおよびポート番号を検索する

キャッシュノードに接続するには、アプリケーションがそのノードのエンドポイントとポート番号を認識している必要があります。

ノードのエンドポイントおよびポート番号を検索する (コンソール)

ノードエンドポイントとポート番号を調べるには

1. Amazon ElastiCache 管理コンソールにサインインし、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すべてのクラスターが一覧表示されます。
2. 実行しているエンジンや設定に対して、以下を行います。
3. 対象のクラスターの名前を選択します。
4. 関心があるノードの [ポート] および [エンドポイント] 列を見つけます。

キャッシュノードのエンドポイントおよびポート番号を検索する (AWS CLI)

キャッシュノードのエンドポイントとポート番号を確認するには、`describe-cache-clusters` コマンドを `--show-cache-node-info` パラメータを指定して使用します。

```
aws elasticache describe-cache-clusters --show-cache-node-info
```

完全修飾DNS名とポート番号は、出力のエンドポイントセクションにあります。

キャッシュノードエンドポイントとポート番号の検索 (ElastiCache API)

キャッシュノードのエンドポイントとポート番号を確認するには、`DescribeCacheClusters` アクションを `ShowCacheNodeInfo=true` パラメータを指定して使用します。

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
```

```
    &Version=2014-09-30
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20140421T220302Z
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

自動検出を使用するための接続

アプリケーションが自動検出を使用する場合、調べる必要があるのは各キャッシュノードの個々のエンドポイントではなく、クラスターの設定エンドポイントだけです。詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。

> **Note**
>
> 現在のところ、自動検出は Memcached エンジンを実行しているキャッシュクラスターでのみ使用できます。

Valkey または Redis OSSクラスター内のノードへの接続

> **Note**
>
> 現時点では、レプリケーションとリードレプリカをサポートするクラスター (API/CLI: レプリケーショングループ) は、Valkey または Redis を実行しているクラスターでのみサポートされていますOSS。

クラスターの場合、 はコンソール、CLI、およびAPIインターフェイス ElastiCache を提供し、個々のノードの接続情報を取得します。

読み取り専用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。ただし、書き込みアクティビティでは、アプリケーションがノードに直接接続するのではなく、クラスターのプライマリエンドポイント (バルキーまたは Redis OSS (クラスターモードが無効)) または設定エンドポイント (バルキーまたは Redis OSS (クラスターモードが有効)) に接続することをお勧めします。これにより、リードレプリカをプライマリロールに昇格させることでクラスターを再設定することにした場合でも、アプリケーションは常に正しいノードを見つけることができます。

レプリケーショングループのクラスターへの接続 (コンソール)

エンドポイントとポート番号を調べるには

- トピックを参照してください。Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)

レプリケーショングループのクラスターへの接続 (AWS CLI)

キャッシュノードのエンドポイントとポート番号を調べるには

レプリケーショングループの名前を指定して、describe-replication-groups コマンドを使用します。

```
aws elasticache describe-replication-groups redis2x2
```

このコマンドでは、次のような出力が生成されます。

```
{
    "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": "redis2x2-0001-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0001-002"
                        }
                    ]
                },
                {
```

```
                    "Status": "available",
                    "Slots": "8192-16383",
                    "NodeGroupId": "0002",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2b",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0002-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0002-002"
                        }
                    ]
                }
            ],
            "ConfigurationEndpoint": {
                "Port": 6379,
                "Address": "redis2x2.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
            },
            "ClusterEnabled": true,
            "ReplicationGroupId": "redis2x2",
            "SnapshotRetentionLimit": 1,
            "AutomaticFailover": "enabled",
            "SnapshotWindow": "13:00-14:00",
            "MemberClusters": [
                "redis2x2-0001-001",
                "redis2x2-0001-002",
                "redis2x2-0002-001",
                "redis2x2-0002-002"
            ],
            "CacheNodeType": "cache.m3.medium",
            "PendingModifiedValues": {}
        }
    ]
}
```

レプリケーショングループのクラスターへの接続 (ElastiCache API)

キャッシュノードのエンドポイントとポート番号を調べるには

以下のパラメータを使って DescribeReplicationGroups を呼び出します。

ReplicationGroupId = レプリケーショングループの名前。

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ReplicationGroupId=repgroup01
    &Version=2014-09-30
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20140421T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

DNS 名前と基盤となる IP

クライアントには、キャッシュデータが保存されているサーバーのアドレスとポートが含まれるサーバーリストが保持されています。を使用する場合 ElastiCache、 DescribeCacheClusters API (または describe-cache-clustersコマンドラインユーティリティ) は、サーバーリストに使用できる完全修飾DNSエントリとポート番号を返します。

> **Important**
>
> クライアントアプリケーションがキャッシュノードエンドポイントに接続しようとするときに、キャッシュノードDNSの名前を頻繁に解決するように設定することが重要です。

VPC インストール

ElastiCache は、障害発生時にキャッシュノードが復旧されたときに、キャッシュノードDNSの名前と IP アドレスの両方が同じであるようにします。

インストールVPC以外

ElastiCache は、障害発生時にキャッシュノードが復旧されたときにキャッシュノードDNSの名前が変更されないようにします。ただし、キャッシュノードの基盤となる IP アドレスは変更される可能性があります。

ほとんどのクライアントライブラリは、永続的なキャッシュノード接続をデフォルトでサポートします。を使用する場合は、永続的なキャッシュノード接続を使用することをお勧めします ElastiCache。クライアント側のDNSキャッシュは、クライアントライブラリ、言語ランタイム、クライアントオペレーティングシステムなど、複数の場所で発生する可能性があります。各レイヤーのアプリケーション設定を確認して、キャッシュノードの IP アドレスを頻繁に解決するようにしてください。

のデータ階層化 ElastiCache

ElastiCache レプリケーショングループで構成され、r6gd ファミリーのノードタイプを使用する Valkey または Redis OSSクラスターでは、そのデータ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) を使用することで、Valkey または Redis OSSワークロードに新しい価格パフォーマンス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、 上のデータにアクセスする際に追加の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SSD。

データ階層化を持つ ElastiCache クラスターでは、 は、保存するすべてのアイテムの最後のアクセス時間 ElastiCache を監視します。使用可能なメモリ (DRAM) が完全に消費されると、 ElastiCache は最も最近使用された (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 に自動的に移動しますSSD。のデータがSSDその後アクセスされると、リクエストを処理する前に ElastiCache 、自動的かつ非同期でメモリに戻ります。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、 はメモリとディスクの値の配置LRUを管理します。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仮定すると、メモリ内のデータへのリクエストSSDと比較して、 に保存されているデータへのリクエストには平均して 300 マイクロ秒のレイテンシーが追加で発生すると予想されます。

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、 でサポートされているすべての Valkey または Redis OSS 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

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、 上のデータにアクセスするときに追加の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 SSD。
- データ階層型ノードで使用可能なSSD容量を使用する場合は、値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。項目が DRAMと の間で移動されるとSSD、キーは常にメモリに残り、値のみがSSD階層に移動します。

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。
- Valkey 7.2 以降のエンジン、または Redis 6.2 OSS 以降のエンジンを使用する必要があります。
- 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、「スケーリング ElastiCache」を参照してください。
- 自動スケーリングは、Valkey バージョン 7.2 以降、および Redis OSS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、「Auto Scaling Valkey クラスターと Redis OSSクラスター」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。
- フォークレスセーブは、Valkey バージョン 7.2 以降、および Redis OSS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 に移動されませんSSD。

料金

R6gd ノードの合計容量 (メモリ + SSD) は 4.8 倍であり、R6g ノード (メモリのみ) と比較して最大使用率で実行した場合、60% を超える削減を実現できます。詳細については、ElastiCache 「 の料金」を参照してください。

モニタリング

ElastiCache には、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が用意されています。DRAM と比較した の項目の比率をモニタリングするにはSSD、 `CurrItems`のメトリクスを Valkey と Redis のメトリクスOSSで使用できます。パーセンテージは、 (ディメンション: 階層 = メモリ * 100 CurrItems の場合) / (ディメンションフィルターCurrItems なしの場合) 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% を下回ると、ElastiCache は項目のエビクションを開始します。逸脱ポリシーで設定されたノードでは、書き込みオペレーションはメモリ不足エラーを受け取ります。

メモリ内の項目の割合が 5% を下回る場合は、クラスターモードが有効になっている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になっている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。スケーリングの詳細については、「」を参照してくださいValkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効) 。データ階層化を使用する Valkey または Redis OSS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Valkey と Redis のメトリクス OSS。

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、Valkey または Redis 用のクラスターの作成 OSS 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ときは AWS CLI、cash.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 OSS 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
```

Windows の場合:

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
   --replication-group-description "Redis OSS 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 OSS 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 (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ElastiCache API。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) — Valkey または Redis OSS (クラスターモードが有効) 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. シャードの数 – 新しいクラスター (API/CLI: ノードグループ) に必要なシャードの数を選択します。
    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ー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. を選択 VPC — このクラスターを作成する VPC を選択します。
    10. パラメータグループ – 選択したノードタイプの Valkey または Redis オーバーOSS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、Valkey または Redis 用のクラスターの作成 OSS 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で使用されます。

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 OSS 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 OSS 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 OSS 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
```

```
    }
}
```

でのクラスターの準備 ElastiCache

次に、 ElastiCache コンソール、、 AWS CLIまたは を使用してクラスターを作成する手順を示します ElastiCache API。

を使用して ElastiCache クラスターを作成することもできますAWS CloudFormation。詳細については、AWS Cloud Formation ユーザーガイド のAWS「：ElastiCache:::CacheCluster」を参照してください。これには、そのアプローチを実装する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

- ElastiCache クラスター要件の決定
- ノードサイズの選択

ElastiCache クラスター要件の決定

準備

以下の質問に対する回答を知ることで、 ElastiCache クラスターの作成をよりスムーズに行うことができます。

- 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 に基づく仮想プライベートクラウド (VPC) でクラスターを起動しますかVPC？

> **Important**
>
> でクラスターを起動する場合はVPC、クラスターの作成VPCを開始する前に、必ず同じ にサブネットグループを作成してください。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 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EC2。ただし、Amazon VPCに基づいて で を起動VPCし、クラスターが にある場

合はVPC、 の外部からアクセスを提供できます AWS。詳細については、「外部から ElastiCache リソースにアクセスする AWS」を参照してください。

- パラメーター値をカスタマイズする必要がありますか。

  その場合、カスタムパラメータグループを作成します。詳細については、「 ElastiCache パラメータグループの作成」を参照してください。

  Valkey または Redis を実行している場合はOSS、 `reserved-memory` または の設定を検討してくださいreserved-memory-percent。詳細については、「Valkey と Redis の予約済みメモリの管理 OSS」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、「 のセキュリティVPC」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

- ElastiCache メモリとプロセッサの要件
- Memcached クラスターの構成
- Valkey クラスターと Redis OSSクラスターの設定
- ElastiCache スケーリング要件
- ElastiCache アクセス要件
- のリージョン、アベイラビリティーゾーン、ローカルゾーンの要件 ElastiCache

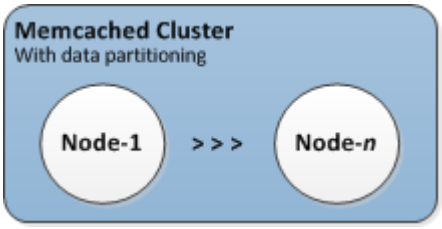
ElastiCache メモリとプロセッサの要件

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。

Memcached エンジンはマルチスレッドであるため、ノードのコア数がクラスターで利用可能な処理能力に影響します。

Memcached クラスターの構成

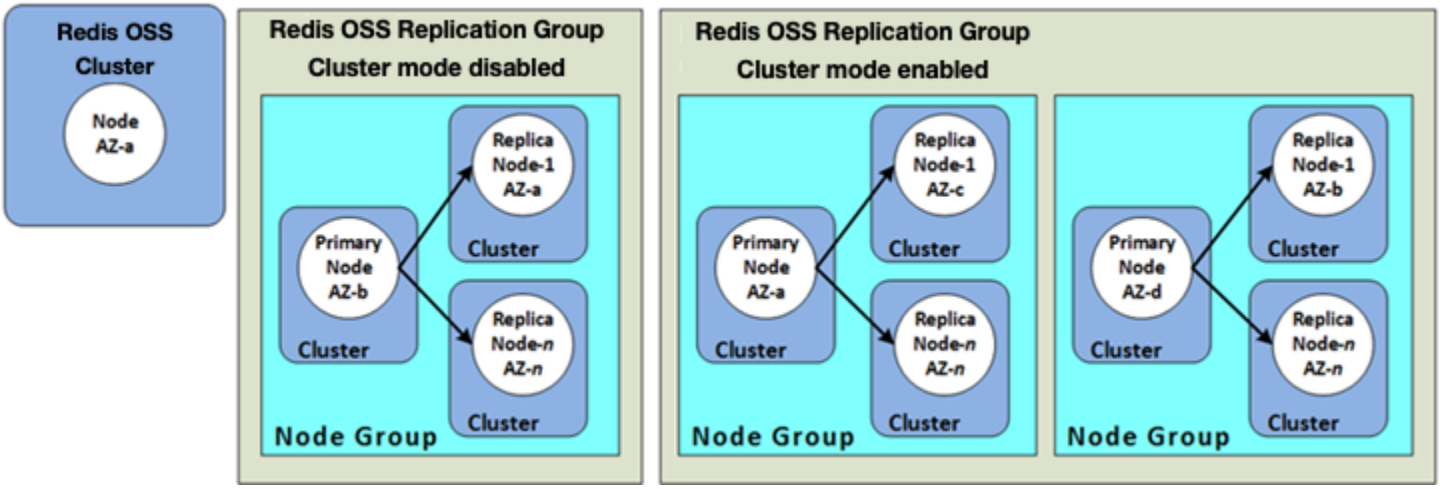
ElastiCache (Memcached) クラスターは 1～60 ノードで構成されます。Memcached クラスター内のデータは、クラスター内のノード間で分割されます。アプリケーションは、エンドポイントと呼ばれるネットワークアドレスを使用して Memcached クラスターに接続します。Memcached クラスター内の各ノードには固有のエンドポイントがあり、アプリケーションはこれを使用して特定のノードに対して読み取りと書き込みを行います。ノードエンドポイントに加えて、Memcached クラスター自体には設定エンドポイントと呼ばれるエンドポイントがあります。アプリケーションはこのエンドポイントを使用してクラスターの読み取りまたは書き込みを行うことができ、どのノードに対して読み取りまたは書き込みを行うかの判断は 自動検出 に任せることができます。



詳細については、「でのクラスターの管理 ElastiCache」を参照してください。

Valkey クラスターと Redis OSSクラスターの設定

ElastiCache と Valkey クラスターと Redis OSS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。Valkey または Redis OSSクラスター内のデータは、クラスター内のシャード間でパーティション化されます。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 Valkey または Redis OSSクラスターに接続します。Valkey または Redis OSS シャードのノードは、1 つの読み取り/書き込みプライマリと他のすべてのノードの読み取り専用セカンダリ (リードレプリカとも呼ばれます) の 2 つのロールのいずれかを満たします。ノードエンドポイントに加えて、Valkey または Redis OSS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との間で読み取りまたは書き込みを行い、どのノードから読み取りまたは書き込みを最大 ElastiCache (Redis) にするかを決定できますOSS。



詳細については、「でのクラスターの管理 ElastiCache」を参照してください。

ElastiCache スケーリング要件

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。Memcached クラスターをスケールアップすると、新しいクラスターは空から起動します。Valkey または Redis OSSクラスターをスケールアップするときは、バックアップからシードして、新しいクラスターが空にならないようにすることができます。

Amazon ElastiCache for Memcached クラスターはスケールアウトまたはスケールインできます。Memcached クラスターをスケールアウトまたはスケールインするには、単純にクラスターにノードを追加したり、クラスターからノードを削除します。自動検出を有効にし、アプリケーションがクラスターの設定エンドポイントに接続している場合は、ノードの追加または削除時にアプリケーションに変更を加える必要はありません。

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

ElastiCache アクセス要件

設計上、Amazon ElastiCache クラスターは Amazon EC2インスタンスからアクセスされます。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アクセスする前に、Amazon EC2インスタンスがクラスターにアクセスすることを承認する必要があります。これを行う手順は、 を EC2-VPCClassic に起動したか、 EC2-Classic に起動したかによって異なります。

クラスターを に起動した場合EC2、VPC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。クラスターを EC2-Classic に起動した場合は、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を許可す

る必要があります。詳細な手順については、このガイドの「ステップ 3. クラスターへのアクセスを許可する」を参照してください。

のリージョン、アベイラビリティーゾーン、ローカルゾーンの要件 ElastiCache

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、レイテンシーを減らすことができます。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。

- のリージョンとアベイラビリティーゾーンの選択 ElastiCache
- でのローカルゾーンの使用 ElastiCache
- 障害の軽減

ノードサイズの選択

ElastiCache クラスター用に選択したノードサイズは、コスト、パフォーマンス、耐障害性に影響します。

ノードサイズ (Valkey と Redis OSS)

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。

以下の質問に回答すると、Valkey または Redis のOSS実装に必要な最小ノードタイプを判断するのに役立ちます。

- 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 OSSバージョン 5.0.6 以降を実行している場合は、Redis OSS エンジンの代わりにクライアント接続のオフロードに使用可能な拡張 I/O 機能CPUsを使用すると、スループットとレイテンシーが向上します。Redis OSSバージョン 7.0.4 以降を実行している場合は、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。各専用ネットワーク IO スレッドは、複数のクライアントから Redis OSS エンジンにコマンドをパイプラインし、Redis OSSの 機能を利用してコマンドをバッチで効率的に処理します。 ElastiCache (Redis OSS) v7.1 以降では、拡張 I/O スレッド機能を拡張して、プレゼンテーションレイヤーロジックも処理しました。プレゼンテーションレイヤーとは、拡張 I/O スレッドがクライアント入力を読み取るだけでなく、入力を Redis OSSバイナリコマンド形式に解析し、実行のためにメインスレッドに転送してパフォーマン

スを向上させるということです。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。

  この場合、Redis OSS エンジンバージョン 6.2 以降で実行している場合は、r6gd ノードタイプを選択してデータ階層化を活用できます。データ階層化では、最近使用されていないデータは に保存されますSSD。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、「のデータ階層化 ElastiCache」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。

- データに必要となる合計メモリ量。

  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。

- どのバージョンの Redis を実行OSSしていますか？

  2.8.22 より前の Redis OSSバージョンでは、フェイルオーバー、スナップショット、同期、レプリカのプライマリオペレーションへの昇格のためにより多くのメモリを予約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 OSSバージョン 2.8.22 以降では、以前のプロセスよりも少ないメモリを必要とするフォークレス保存プロセスを使用します。

  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
  - Valkey または Redis OSS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を確認する

- 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。例としては、スナップショットを作成する場合、

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する場合、および追加専用ファイル (AOF) 機能を有効にする場合などがあります。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、「Valkey または Redis OSS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を確認する」を参照してください。

- 実装は、スタンドアロンの Valkey または Redis OSS (クラスターモードが無効) クラスターですか、それとも複数のシャードを持つ Valkey または Redis OSS (クラスターモードが有効) クラスターですか？

  Valkey または Redis OSS (クラスターモードが無効) クラスター

  Valkey または Redis OSS (クラスターモードが無効) クラスターを実装する場合、ノードタイプは、前の箇条書きで説明したように、すべてのデータに加えて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 Valkey または Redis OSS (クラスターモードが有効)

  複数のシャードを持つ Valkey または Redis OSS (クラスターモードが有効) クラスターを実装する場合、ノードタイプは`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?

ローカルゾーンを使用すると、 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。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`

クラスターの実行中に、メモリ使用量、プロセッサ使用率、キャッシュヒット、キャッシュミスのメトリクスをモニタリングできます。メトリクスは に発行されます CloudWatch。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。このような場合は、より大きな CPU とメモリ仕様を持つ別のノードサイズを選択できます。

CPU 使用状況をモニタリングするときは、Valkey と Redis OSSが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、報告されたCPU使用量にCPUコア数を掛けて、実際の使用量を取得します。例えば、使用率が 20% とCPU報告される 4 コアは、実際には Redis が 80% の使用率で実行OSSしている 1 つのコアです。

ノードサイズ (Memcached)

Memcached クラスターには 1 つまたは複数のノードが含まれ、クラスターのデータは複数のノードに分割されます。このため、クラスターで必要なメモリとノードで必要なメモリには関連性がありますが、同じではありません。ノードの数を減らすか、より多くのより小さなノードを確保することにより、必要なクラスターメモリの容量を実現できます。また、ニーズの変化に応じてクラスターへのノードの追加や削除を行い、必要なものだけに支払うことができます。

クラスターの合計メモリ容量は、システムオーバーヘッドを差し引いた後に、クラスター内のノード数に各ノードのRAM容量を掛けて計算されます。各ノードの容量はノードタイプに基づいています。

```
cluster_capacity = number_of_nodes * (node_capacity - system_overhead)
```

クラスター内のノード数は、Memcached を実行するクラスターの可用性の重要な要素です。1 つのキャッシュノードで障害が発生した場合、アプリケーションの可用性とバックエンドデータベースの負荷に影響を与える可能性があります。このような場合、 ElastiCache は障害が発生したノードの代替をプロビジョニングし、再入力されます。この可用性への影響を小さくするには、使用している大容量のノードを減らすのではなく、メモリとコンピューティングの容量をより小さい容量のより多くのノードに分散させます。

35 GB のキャッシュメモリが必要なシナリオでは、以下のいずれかを設定できます。

- それぞれ 3.22 GB のメモリと 2 つのスレッドを持つ 11 の `cache.t2.medium` ノード = 35.42 GB、22 スレッド。
- それぞれ 6.42 GB のメモリと 2 つのスレッドを持つ 6 つの `cache.m4.large` ノード = 38.52 GB、12 スレッド。
- それぞれ 12.3 GB のメモリと 2 つのスレッドを持つ 3 つの `cache.r4.large` ノード = 36.90 GB、6 スレッド。
- それぞれ 14.28 GB のメモリと 4 つのスレッドを持つ 3 つの `cache.m4.xlarge` ノード = 42.84 GB、12 スレッド。

ノードオプションの比較

ノードの種類	メモリ (GiB)	コア	時間あたりのコスト*	必要なノード	合計メモリ (GiB)	合計コア	毎月のコスト
cache.t2.medium	3.22	2	0.068 USD	11	35.42	22	538.56 USD
cache.m4.large	6.42	2	0.156 USD	6	38.52	12	673.92 USD
cache.m4.xlarge	14.28	4	0.311 USD	3	42.84	12	$ 671.76
cache.m5.xlarge	12.93	4	0.311 USD	3	38.81	12	$ 671.76
cache.m6g.large	6.85	2	$ 0.147	6	41.1	12	635 ドル

ノードの種類	メモリ (GiB)	コア	時間あたりのコスト*	必要なノード	合計メモリ (GiB)	合計コア	毎月のコスト
cache.r4.large	12.3	2	0.228 USD	3	36.9	6	$ 492.48
cache.r5.large	13.07	2	0.216 USD	3	39.22	6	466.56 USD
cache.r6g.large	13.07	2	$ 0.205	3	42.12	6	442 ドル

* 2020 年 10 月 8 日現在のノードあたりの時間あたりのコスト。

30 日 (720 時間) の使用率 100% の場合の 1 か月あたりのコスト。

これらのオプションは、それぞれ同様のメモリ容量を提供しますが、コンピューティング容量とコストは異なります。特定のオプションのコストを比較するには、「Amazon ElastiCache 料金」を参照してください。

Memcached を実行するクラスターでは、各ノードの使用可能なメモリの一部は接続のオーバーヘッド用に使用されます。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。

複数のノードを使用して、それらの間でキーを分散させる必要があります。各ノードには、独自のエンドポイントがあります。エンドポイント管理を容易にするために、 ElastiCache Auto Discovery 機能を使用できます。この機能を使用すると、クライアントプログラムはクラスター内のすべてのノードを自動的に識別できます。詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。

場合によっては、必要な容量がわからないことがあります。その場合、テストのために、1 つの `cache.m5.large` ノードから始めることをお勧めします。次に、Amazon に公開されている ElastiCache メトリクスを使用して、メモリ使用量、CPU使用率、キャッシュヒット率をモニタリングします CloudWatch。の CloudWatch メトリクスの詳細については ElastiCache、「」を参照してください CloudWatch メトリクスの使用のモニタリング。本番稼働ワークロードや大規模なワークロードの場合、R5 ノードは最高のパフォーマンスとRAMコスト価値を提供します。

クラスターに必要なヒット率がない場合は、ノードを簡単に追加して、クラスターで使用可能なメモリの合計を増やすことができます。

クラスターが にバインドされているCPUが、十分なヒット率がある場合は、より多くのコンピューティング能力を提供するノードタイプで新しいクラスターを設定します。

Valkey または Redis 用のクラスターの作成 OSS

次の例は、 AWS Management Console、、 AWS CLI および を使用して Valkey または Redis OSS クラスターを作成する方法を示しています ElastiCache API。

Valkey または Redis の作成 OSS (クラスターモードが無効) (コンソール)

ElastiCache Valkey または Redis OSS エンジンを使用する場合、 はレプリケーションをサポートします。データが Valkey または Redis OSSの読み取り/書き込みプライマリクラスターに書き込まれるときと、読み取り専用のセカンダリクラスターに伝播されるとき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、現在のユニバーサルタイム (UTC) 時間です。Valkey または Redis OSSクラスターは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場合でも、すべての Valkey または Redis OSS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。

Valkey または Redis OSS (クラスターモードが無効) クラスターを作成するには、「」のステップに従いますValkey (クラスターモードが無効) クラスターの作成 (コンソール) 。

クラスターのステータスが利用可能になるとすぐに、クラスターへのアクセスを Amazon に許可し、クラスターに接続して使用を開始できます。 EC2詳細については、「ステップ 3. クラスターへのアクセスを許可する」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。「でのクラスターの削除 ElastiCache」を参照してください。

Valkey または Redis OSS (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 OSS3.2.4 以降を実行している場合は、Valkey または Redis OSS (クラスターモードが有効) クラスターを作成できます。Valkey または Redis OSS (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) にデータを分割できますが、制限があります。Valkey または Redis OSS (クラスターモードが無効) と Valkey または Redis OSS (クラスター

モードが有効) の比較については、「」を参照してくださいサポートされているエンジンとバージョン。

ElastiCache コンソールを使用して Valkey または Redis OSS (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、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 右上隅のリストから、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. Create VPC を選択し、「仮想プライベートクラウドの作成 (VPC)」で説明されている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、バルキークラスターの作成 または Redis OSS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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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、「Valkey パラメータと Redis

OSSパラメータ」および「ElastiCache パラメータグループの作成」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、「のデータ階層化 ElastiCache」を参照してください。

e. シャードの数 で、この Valkey または 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 OSS (パーティション/ノードグループ) の数を選択します。

   Valkey または Redis の一部のバージョン OSS (クラスターモードが有効) では、クラスター内のシャードの数を動的に変更できます。

   - Redis OSS3.2.10 以降 – クラスターが Redis OSS3.2.10 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の数を動的に変更できます。詳細については、「Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)」を参照してください。
   - その他の Redis OSSバージョン – クラスターがバージョン 3.2.10 OSSより前のバージョンの Redis を実行している場合は、別のアプローチがあります。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。

Valkey または Redis OSS (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) あたりのレプリカの数が増減する場合は、新しい数のレプリカ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、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」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 の場合、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ために IPv4 と IPv6 アドレスの両方が割り当てられているデュアルスタックサブネットと、 IPv6IPv6のみのサブネットが動作するように割り当てられているデュアルスタック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、それが属する 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。

   - でネットワークタイプを選択する ElastiCache.
   - にサブネットを作成しますVPC。

   での ローカルゾーンの使用 ElastiCache 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] (アベイラビリティーゾーンの配置) には 2 つのオプションがあります。

   - 設定なし – アベイラビリティーゾーン ElastiCache を選択します。
   - [アベイラビリティーゾーンの指定] – 各クラスターに対するアベイラビリティーゾーンを指定します。

     アベイラビリティー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ーゾーンを選択します。

   詳細については、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」を参照してください。

5. [Next] (次へ) を選択します。
6. Advanced Valkey 設定または Advanced Redis OSS設定、または

   - [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できます。詳細については、「 の AWS カスタマーマネージドキーの使用KMS」を参照してください。

     - [転送時の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「転送中の暗号化」を参照してください。Valkey 7.2 以降または Redis OSS 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、次のいずれかのアクセスコントロールオプションを指定するように求められます。

       - 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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。

- AUTH デフォルトユーザー – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

- AUTH – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

> **Note**
>
> OSS バージョン 3.2.10 を除く 3.2.6 以降の Redis バージョンでは、 が唯一のオプションAUTHです。

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。のデフォルトのセキュリティグループを使用するVPCか、新しいセキュリティグループを作成できます。

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

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。

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

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、クラスターのシステムメンテナンスをスケジュールする ElastiCache毎週の時間で、通常は 1 時間です。は ElastiCache、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。すべての時間は UCT 時間です。

詳細については、「ElastiCache クラスター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]:
   - ログ形式 で、テキストまたは を選択しますJSON。
   - 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。
10. 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、タグの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 ElastiCache (Redis OSS) でのダウンタイムの最小化」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 コンソール AWS CLI の代わりに または ElastiCache APIを使用して同等の 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利用可能になるとすぐに、EC2クラスターへのアクセスを許可し、クラスターに接続して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を許可する」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。「でのクラスターの削除 ElastiCache」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。「でのクラスターの削除 ElastiCache」を参照してください。

Valkey または Redis OSS (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 Valkey または Redis OSS (クラスターモードが無効) クラスター

次のCLIコードは、レプリカなしで Valkey または Redis OSS (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
```

Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (AWS CLI)

Valkey または Redis OSS (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。Valkey または Redis OSS (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、「」を参照してくださいValkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成する (AWS CLI) 。

詳細については、 ElastiCache 「 リファレンストピック AWS CLI 」を参照してくださいcreate-replication-group。

Valkey または Redis 用のクラスターの作成 OSS (ElastiCache API)

を使用してクラスターを作成するには ElastiCache API、 `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。「でのクラスターの削除 ElastiCache」を参照してください。

トピック

- Valkey または Redis OSS (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)
- Valkey または Redis でのキャッシュクラスターの作成 OSS (クラスターモードが有効) (ElastiCache API)

Valkey または Redis OSS (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、Valkey または Redis OSS (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ElastiCache API。

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>
```

Valkey または Redis でのキャッシュクラスターの作成 OSS (クラスターモードが有効) (ElastiCache API)

Valkey または Redis OSS (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。Valkey または Redis OSS (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、「」を参照してくださいValkey または Redis OSS (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ゼロから作成する (ElastiCache API) 。

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

Memcached 用のクラスターの作成

次の例は、 AWS Management Console、、 AWS CLI および を使用してクラスターを作成する方法を示しています ElastiCache API。

Memcached クラスター (CLI) の作成 (コンソール)

Memcached エンジンを使用すると、Amazon ElastiCache は複数のノードにデータを水平にパーティション分割できます。Memcached によって自動検出が有効になるため、各ノードのエンドポイントを手動で追跡する必要はなくなります。Memcached によって各ノードのエンドポイントは追跡されて、ノードの追加と削除に応じてエンドポイントのリストが更新されます。アプリケーションとクラスターのやり取りで必要になるのは、設定エンドポイントのみになります。

Memcached クラスターを作成するには、「クラスターを作成する」の手順に従います。

クラスターのステータスが利用可能になるとすぐに、クラスターへのアクセスを Amazon に許可し、クラスターに接続して使用を開始できます。 EC2詳細については、「ステップ 3. クラスターへのアクセスを許可する」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。「でのクラスターの削除 ElastiCache」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。「でのクラスターの削除 ElastiCache」を参照してください。

Memcached キャッシュクラスターの作成 (AWS CLI)

次のCLIコードは、3 つのノードを持つ Memcached 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 memcached \
--engine-version 1.4.24 \
--cache-parameter-group default.memcached1.4 \
--num-cache-nodes 3
```

Windows の場合:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine memcached ^
--engine-version 1.4.24 ^
--cache-parameter-group default.memcached1.4 ^
--num-cache-nodes 3
```

Memcached 用のクラスターの作成 (ElastiCache API)

を使用してクラスターを作成するには ElastiCache API、 `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。「でのクラスターの削除 ElastiCache」を参照してください。

トピック

- Memcached キャッシュクラスターの作成 (ElastiCache API)

Memcached キャッシュクラスターの作成 (ElastiCache API)

次のコードは、3 つのノード () を持つ Memcached クラスターを作成しますElastiCache API。

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
    &Engine=memcached
    &NumCacheNodes=3
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &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>
```

ElastiCache クラスターの詳細の表示

ElastiCache コンソール、、または を使用して AWS CLI、1 つ以上のクラスターに関する詳細情報を表示できます ElastiCache API。

Memcached クラスターの詳細の表示 (コンソール)

Memcached クラスターの詳細は、 ElastiCache コンソール、 AWS CLI for ElastiCache、または を使用して表示できます ElastiCache API。

次の手順では、 ElastiCache コンソールを使用して Memcached クラスターの詳細を表示する方法について説明します。

Memcached クラスターの詳細を表示するには

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 右上隅のリストから、関心のある AWS リージョンを選択します。
3. ElastiCache エンジンダッシュボードで、Memcached を選択します。これにより、Memcached エンジンで実行されているすべてのクラスターのリストが表示されます。
4. 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。
5. ノード情報を表示するには、[Nodes] (ノード) タブを選択します。このタブには、ノードの状態とエンドポイントに関する情報が表示されます。
6. 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 メトリクスの使用のモニタリング」を参照してください
7. Network and Security 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、およびVPCセキュリティグループ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、「 ElastiCache クラスターメンテナンスの管理」を参照してください。
9. [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。

Valkey または Redis OSS (クラスターモードが無効) の詳細の表示 (コンソール)

Valkey または Redis OSS (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、ElastiCache コンソール、 AWS CLI for ElastiCache、または を使用します ElastiCache API。

次の手順では、 ElastiCache コンソールを使用して Valkey または Redis OSS (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について説明します。

Valkey または Redis OSS (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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ElastiCache エンジンダッシュボードで、Valkey または Redis OSS のいずれかを選択して、そのエンジンで実行されているすべてのクラスターのリストを表示します。
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。Valkey または Redis OSS エンジンを実行しているクラスターを選択し、Clustered Valkey または Clustered Redis は選択しないでくださいOSS。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、「 ElastiCache クラスター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、「のサービス更新 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。

Valkey または Redis OSS (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)

Valkey または Redis OSS (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、ElastiCache コンソール、 AWS CLI for ElastiCache、または を使用します ElastiCache API。

次の手順では、 ElastiCache コンソールを使用して Valkey または Redis OSS (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について説明します。

Valkey または Redis OSS (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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 右上隅のリストから、関心のある AWS リージョンを選択します。
3. ElastiCache エンジンダッシュボードで、Valkey または Redis OSS を選択して、そのエンジンで実行されているすべてのクラスターのリストを表示します。
4. Valkey または Redis OSS (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。Valkey または Clustered Redis OSS エンジンを実行しているクラスターを選択してください。

   クラスターの下の画面が展開され、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. 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の設定、VPCセキュリティグループ、およびクラスターで有効になっている暗号化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 のデータセキュリティ ElastiCache」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、「 ElastiCache クラスター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、「のサービス更新 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。

ElastiCache クラスターの詳細の表示 (AWS CLI)

次のコードは、 の詳細を一覧表示します。*my-cluster*:

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

置換 *my-cluster* create-cache-cluster コマンドを使用して 1 つのキャッシュノードと 0 つのシャードでクラスターが作成されている場合、クラスターの名前。

```
{
    "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
        }
    ]
```

```
}
```

を使用してクラスターを作成する場合 AWS Management Console (1 つ以上のシャードでクラスターノードを有効または無効にする場合) 、次のコマンドを使用してクラスターの詳細を記述します (置き換える *my-cluster* レプリケーショングループの名前 (クラスターの名前):

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

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

ElastiCache クラスターの詳細の表示 (ElastiCache API)

DescribeCacheClusters アクションを使用して ElastiCache API、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、「 リファレンストピック ElastiCache API」を参照してくださ
いDescribeCacheClusters。

ElastiCache クラスターの変更

ElastiCache クラスターからノードを追加または削除することに加えて、セキュリティグループの追加、メンテナンスウィンドウの変更、パラメータグループの変更など、他の変更が必要になる場合があります。

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。特定のパラメータ変更が適用されるタイミングを確認するには、 Memcached 固有のパラメータおよび のテーブルの詳細列の「Changes Take Effect」セクションを参照してくださいValkey パラメータと Redis OSSパラメータ。クラスターのノードの再起動については、「ノードの再起動」を参照してください。

の使用 ElastiCache 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、Valkey または Redis OSSクライアントは、クラスターモードが有効とクラスターモードが無効の両方を使用して接続できます。すべての Valkey または Redis OSSクラ

イアントをクラスターモードが有効になるように移行したら、クラスターモード設定を完了し、クラスターモードを有効 に設定することができます。

- 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、「のバージョン管理 ElastiCache」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 (複数可)
- パラメータグループ
- ノードの種類

> **Note**
>
> クラスターが r6gd ファミリーのノードタイプを使用している場合は、そのファミリー内からのみ別のノードサイズを選択できます。r6gd ファミリーからノードタイプを選択すると、データ階層化が自動的に有効になります。詳細については、データ階層化を参照してください。

- 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

- Memcached エンジンのバージョン互換性
- ネットワークの種類

> **Note**
>
> IPv4 から に切り替える場合はIPv6、 と互換性のあるサブネットグループを選択または作成する必要がありますIPv6。詳細については、「でネットワークタイプを選択する ElastiCache」を参照してください。

- VPC セキュリティグループ (複数可)
- パラメータグループ
- メンテナンスウィンドウ
- 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のみ適用されます。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、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。

Redis のログ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. クラスターの詳細ページで、ログタブを選択します。
3. スローログを有効または無効にするには、 を有効または無効にします。

   有効化を選択した場合:

   a. ログ形式 で、 JSON または 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. Log destination で、Create new を選択し、 CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名のいずれかを入力できます。既存の選択を選択し、CloudWatchLogs ロググループ名または Kinesis Data Firehose ストリーム名のいずれかを選択することもできます。

   d. [Enable (有効化)] を選択します。

Redis の設定を変更するには：

1. Modify を選択します。
2. ログ形式 で、 JSON または 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. Log destination で、Create new 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。

AWS CLI での の使用 ElastiCache

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**
>
> Memcached エンジンの新しいバージョンにアップグレードできます。その設定方法の詳細については、「のバージョン管理 ElastiCache」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。

> **Important**
>
> 新しい Valkey または Redis OSS エンジンバージョンにアップグレードできます。Redis OSS5.0.6 から Redis OSS6.0 など、主要なエンジン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、「のバージョン管理 ElastiCache」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のみ適用されます。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、 AWS CLI ElastiCache 「」トピックの「」を参照してくださいmodify-cache-cluster。

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 ElastiCache API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**
>
> Memcached エンジンの新しいバージョンにアップグレードできます。その設定方法の詳細については、「のバージョン管理 ElastiCache」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。

> **Important**
>
> 新しい Valkey または Redis OSS エンジンバージョンにアップグレードできます。Redis OSS5.0.6 から Redis OSS6.0 など、主要なエンジン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、「のバージョン管理 ElastiCache」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のみ適用されます。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、「 リファレンストピック ElastiCache API」を参照してくださいModifyCacheCluster。

ElastiCache クラスターへのノードの追加

Memcached クラスターにノードを追加すると、クラスターのパーティションの数が増えます。クラスターのパーティションの数を変更するときは、正しいノードにマップされるように、一部のキースペースを再マッピングする必要があります。キースペースを再マッピングすると、クラスターでのキャッシュミスの数が一時的に増えます。詳細については、「効率的なロードバランシングのための ElastiCache クライアントの設定 (Memcached)」を参照してください。

Valkey または Redis OSS (クラスターモードが有効) クラスターを再設定するには、「」を参照してください。 Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)

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。

の使用 ElastiCache AWS Management Console

単一ノードの Valkey または Redis OSS (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になっていないクラスター) にノードを追加する場合は、まずレプリケーションを追加し、次にレプリカノードを追加するという 2 つのステップのプロセスです。

トピック

- シャードなしで Valkey または Redis OSSクラスターにレプリケーションを追加するには
- ElastiCache クラスターにノードを追加するには (コンソール)

次の手順では、レプリケーションが有効になっていOSSない単一ノードの Valkey または 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。

シャードなしで Valkey または Redis OSSクラスターにレプリケーションを追加するには

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から、Valkey クラスターまたは Redis OSSクラスター を選択します。

   エンジンを実行しているクラスターのリストが表示されます。
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。

以下は、レプリケーションが有効になっていない Redis OSSクラスターに当てはまります。

- クラスター Redis OSSではなく、Redis を実行していますOSS。
- 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、ElastiCache クラスターにノードを追加するには (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. [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。

ElastiCache クラスターにノードを追加するには (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. 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、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 Valkey または Redis OSS (クラスターモードが有効) クラスターの場合は、「」を参照してくださいValkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)。

   クラスターがシャードがゼロの Valkey または Redis OSS (クラスターモードが無効) クラスターの場合は、まず でステップを完了しますシャードなしで Valkey または Redis OSSクラスターにレプリケーションを追加するには。
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	削除	削除	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。 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。
シナリオ 2	削除	作成	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。 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。
シナリオ 3	作成	削除	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。 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。

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	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。 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。 **Important** 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。

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 での の使用 ElastiCache

を使用してクラスターにノードを追加するには AWS CLI、次のパラメータ`modify-cache-cluster`で AWS CLI オペレーションを使用します。

- `--cache-cluster-id` ノードを追加する先のキャッシュクラスターの ID。

- `--num-cache-nodes` `--num-cache-nodes` パラメータは、変更の適用後に、このクラスターに必要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、`--num-cache-nodes` 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。この値が現在のノード数より小さい場合、 は パラメータcache-node-ids-to-removeと、クラスターから削除するノードのリスト ElastiCache を求めます。詳細については、「AWS CLI での の使用 ElastiCache」を参照してください。
- `--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 modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --num-cache-nodes 5 \
    --apply-immediately
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --num-cache-nodes 5 ^
    --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) 。

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "my-cluster",
        "PreferredAvailabilityZone": "us-west-2b",
        "ConfigurationEndpoint": {
            "Port": 11211,
            "Address": "rlh-mem000.7alc7bf-example.cfg.usw2.cache.amazonaws.com"
        },
        "CacheSecurityGroups": [],
```

```
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 2,
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 5
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",
         "DataTiering": "disabled",
    }
}
```

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

AWS CLI での の使用 ElastiCache

レプリケーションが有効になっていない既存の Valkey または Redis OSS (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、まず既存のクラスターをプライマリとして指定するレプリケーショングループを作成する必要があります。詳細については、「使用可能な Valkey または Redis OSS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。

を使用してクラスターにノードを追加するには AWS CLI、次のパラメータincrease-replica-countで AWS CLI オペレーションを使用します。

- --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 「」トピックを参照してくださいincrease-replica-count。

の使用 ElastiCache API

レプリケーションが有効になっていない既存の Valkey または Redis OSS (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、まず既存のクラスターをプライマリとして指定するレプリケーショングループを作成する必要があります。詳細については、「スタンドアロンの Valkey または Redis OSS (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。

クラスターにノードを追加するには (ElastiCache API)

- 次のパラメータを使用して IncreaseReplicaCountAPI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、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要があります。この値が現在のノード数より小さい場合は、 をノード数DecreaseReplicaCountAPIとともに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、「」トピックを参照してください ElastiCache API[IncreaseReplicaCount](#)。

の使用 ElastiCache API

クラスターにノードを追加するには (ElastiCache API)

- 次のパラメータを使用して ModifyCacheClusterAPIオペレーションを呼び出します。
  - CacheClusterId ノードを追加する先のクラスターの ID。
  - NumCacheNodes NumCachNodes パラメータは、変更の適用後に、このクラスターに必要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、NumCacheNodes 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。この値が現在のノード数より小さい場合、 は、クラスターから削除するノードのリストCacheNodeIdsToRemoveを含む パラメータ ElastiCache を想定します (「」を参照[Memcached での ElastiCache APIの使用](#))。
  - 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=ModifyCacheCluster
    &ApplyImmediately=true
    &NumCacheNodes=5
 &CacheClusterId=my-cluster
 &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、「 トピック」を参照してください ElastiCache API[ModifyCacheCluster]。

ElastiCache クラスターからノードを削除する

Valkey、RedisOSS、または Memcached クラスターからノードを削除するには AWS Management Console、、 AWS CLI、または を使用します ElastiCache API。

> **Note**
>
> Memcached クラスターのノード数を変更するたびに、正しいノードにマップできるようにキースペースの一部を再マッピングする必要があります。Memcached クラスターの負荷分散の詳細については、「効率的なロードバランシングのための ElastiCache クライアントの設定 (Memcached)」を参照してください。

の使用 ElastiCache AWS Management Console

クラスターからノードを削除するには (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. 右上隅のリストから、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 コンソールを使用すると、一度に削除できるノードは 1 つしかないため、複数のノードを選択すると、ノードの削除ボタンを使用できなくなります。

   [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**
>
> Valkey または Redis ではOSS、マルチ AZ に準拠していないクラスターでノード結果を削除する場合は、まずマルチ AZ チェックボックスをオフにしてからノードを削除します。[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	削除	削除	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。 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。
シナリオ 2	削除	作成	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。 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。
シナリオ 3	作成	削除	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。 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。

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	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。 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。 **Important** 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。

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 での の使用 ElastiCache

1. 削除するノードIDsを特定します。詳細については、「ElastiCache クラスターの詳細の表示」を参照してください。
2. 次の例のように、削除するノードのリストで `decrease-replica-count`CLIオペレーションを使用します。

   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IDsするノードのリスト。

- `--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` 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 「」トピックを参照してくださいdecrease-replica-count。

Valkey または Redis での ElastiCache APIの使用 OSS

を使用してノードを削除するには ElastiCache API、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リストを使用して DecreaseReplicaCountAPI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 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、「 トピック」を参照してください ElastiCache APIDecreaseReplicaCount。

Memcached での ElastiCache APIの使用

を使用してノードを削除するには ElastiCache API、次のように、キャッシュクラスター ID と削除するノードのリストを使用して ModifyCacheClusterAPIオペレーションを呼び出します。

- CacheClusterId ノードを削除するキャッシュクラスターの ID。
- NumCacheNodes NumCacheNodes パラメータは、変更の適用後に、このクラスターに必要となるノードの数を指定します。
- CacheNodeIdsToRemove.member.n クラスターから削除IDsするノードのリスト。
  - CacheNodeIdsToRemove.member.1=0004
  - CacheNodeIdsToRemove.member.1=0005

- `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` 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=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &ApplyImmediately=true
    &CacheNodeIdsToRemove.member.1=0004
    &CacheNodeIdsToRemove.member.2=0005
    &NumCacheNodes=3
    &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、「」トピックを参照してください ElastiCache API`ModifyCacheCluster`。

で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 ElastiCache

ElastiCache クラスターの変更をすぐに適用しないことを選択した場合、次のメンテナンスウィンドウで実行されるまで、オペレーションのステータスは保留になります。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. 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、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。

でのクラスターの削除 ElastiCache

ElastiCache クラスターが使用可能な状態にある限り、クラスターをアクティブに使用しているかどうかにかかわらず、そのクラスターに対して課金されます。課金を中止するには、クラスターを削除します。

> **Warning**
>
> ElastiCache クラスターを削除すると、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。

の使用 AWS Management Console

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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ElastiCache エンジンダッシュボードで、削除するクラスターが実行中の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削除できるクラスターは 1 つだけです。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] で、[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 を使用して ElastiCache クラスターを削除する

次のコードは、 ElastiCache キャッシュクラスター を削除しますmy-cluster。

```
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 ElastiCache 「」トピックの「」を参照してくださいdelete-cache-cluster。

の使用 ElastiCache API

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、「 リファレンストピック ElastiCache API」を参照してくださいDeleteCacheCluster。

ElastiCache 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されるように設計されています。

Amazon Virtual Private Cloud (Amazon VPC) で ElastiCache インスタンスを起動した場合、同じ Amazon の Amazon EC2インスタンスからインスタンスにアクセスできます ElastiCacheVPC。または、VPCピアリングを使用して、別の Amazon EC2の Amazon から ElastiCache インスタンスにアクセスできますVPC。

EC2 Classic で ElastiCache インスタンスを起動した場合、EC2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、インスタンスがクラスターにアクセスできるようにします。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

クラスターを EC2- に起動しましたVPC

Amazon Virtual Private Cloud (Amazon VPC) でクラスターを起動した場合、クラスターに接続 ElastiCacheできるのは、同じ Amazon で実行されている Amazon EC2インスタンスからのみですVPC。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はサブネットを他のアベイラビリティーゾーンおよび関連するゲートウェイ、ルートテーブル、その他のセキュリティグループの考慮事項の任意のサブネットとして扱います。 は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. にサインイン AWS Management Console し、 で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。

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 のセキュリティグループを選択しますVPC。ElastiCache 使用するための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の名前はデフォルト 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. タイプ 列で、カスタムTCPルール 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。Memcached のデフォルトポートは **11211** で、Redis のデフォルトポートOSSは **6379**です。
   e. ソースボックスで、ポート範囲 (0.0.0.0/0) がある任意の場所を選択して、Amazon 内で起動する Amazon EC2インスタンスを ElastiCache ノードVPCに接続できるようにします。

      > **Important**
      >
      > ElastiCache クラスターを 0.0.0.0/0 に開いても、クラスターにはパブリック IP アドレスがないため、 の外部からアクセスできないため、インターネットには公開されませんVPC。ただし、デフォルトのセキュリティグループは、顧客のアカウントの他の Amazon EC2インスタンスに適用でき、それらのインスタンスにはパブリック IP アドレスがある場合があります。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。したがって、によってのみ使用されるVPCセキュリティグループを作成することをお勧めしますElastiCache。詳細については、「カスタムセキュリティグループ」を参照してください。

   f. [Save] を選択します。

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

外部から ElastiCache リソースにアクセスする AWS

Amazon ElastiCache は、クラウドベースのメモリ内キーバリューストアを提供する AWS サービスです。このサービスは、 内からのみアクセスされるように設計されています AWS。ただし、ElastiCache クラスターが 内でホストされている場合はVPC、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アクセスを提供できます。

要件

外部から ElastiCache リソースにアクセスするには、次の要件を満たす必要があります AWS。

- クラスターは 内にありVPC、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する必要があります。これは必須の要件であり、例外はありません。
- NAT インスタンスは、クラスターVPCと同じ で起動する必要があります。
- NAT インスタンスは、クラスターとは別のパブリックサブネットで起動する必要があります。
- Elastic IP アドレス (EIP) をNATインスタンスに関連付ける必要があります。iptables のポート転送機能は、NATインスタンス上のポートを 内のキャッシュノードポートに転送するために使用されますVPC。

考慮事項

外部から ElastiCache リソースにアクセスするときは、以下の考慮事項に留意する必要がありますElastiCache。

- クライアントはNATインスタンスの EIPおよびキャッシュポートに接続します。NAT インスタンスでのポート転送は、トラフィックを適切なキャッシュクラスターノードに転送します。
- 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。NAT インスタンス経由でアクセスするキャッシュクラスターの数が増えると、影響が大きくなります。

- 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インスタンス経由で機密データを送信することは避ける必要があります。
- NAT インスタンスは、別のインスタンスを維持するオーバーヘッドを追加します。
- NAT インスタンスは単一の障害点として機能します。NAT で高可用性を設定する方法については VPC、「Amazon VPC NAT インスタンスの高可用性: 例」を参照してください。

外部から ElastiCache リソースにアクセスする方法 AWS

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NATの方向が必要です。

```
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、Memcached 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 Memcached ポート – 11211
  - セキュリティグループ – *10\.0\.0\.55*
- Valkey または Redis OSSクラスターには、次の方法でアクセスします。
  - IP アドレス – 10.0.1.230
  - デフォルトポート – 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 インスタンスには Elastic IP アドレス 203.0.113.73 があります。
- NAT インスタンスにはセキュリティグループ sg-ce56b7a9 があります。

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VPCと同じ にインスタンスを作成しますが、パブリックサブネットに NATインスタンスを作成します。

   デフォルトでは、VPCウィザードは 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。外部 ElastiCache から にアクセスできるようにするEC2NATAMIには、 を使用する必要があります AWS。

   NAT インスタンスの作成の詳細については、「 ユーザーガイド」のNAT「インスタンス」を参照してください。 AWS VPC
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 インスタンスセキュリティグループとクラスターインスタンスには、次のルールが必要です。

   - 2 つのインバウンドルール
     - Memcached では、最初のルールは、信頼できるクライアントTCPからNATインスタンス (11211 ~ 11213) から転送された各キャッシュポートへの接続を許可することです。
     - Valkey と Redis ではOSS、最初のルールは、信頼されたクライアントTCPからNATインスタンス (6379～6381) から転送された各キャッシュポートへの接続を許可することです。
     - 信頼されたクライアントSSHへのアクセスを許可する 2 番目のルール。

   NAT インスタンスセキュリティグループ - Memcached を使用した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	11211-11213	198.51.100.27/32
SSH	TCP	22	198.51.100.27/32

NAT インスタンスセキュリティグループ - Valkey または Redis を使用したインバウンドルール OSS

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

- Memcached では、キャッシュポート (11211) TCPへの接続を許可するアウトバウンドルールです。

  NAT インスタンス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	11211	sg-ce56b7a9 (クラスターインスタンスのセキュリティグループ)

- Valkey または Redis ではOSS、キャッシュポート (6379) TCPへの接続を許可するアウトバウンドルールです。

  NAT インスタンス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)

- Memcached では、NATインスタンスTCPからキャッシュポート (11211) への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です。

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	11211	sg-bd56b7da (NATセキュリティグループ)

- Valkey または Redis ではOSS、NATインスタンスTCPからキャッシュポート (6379) への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です。

  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。

   - 信頼されたクライアントがNATインスタンスSSHに対して にアクセスできることを確認します。
   - 信頼されたクライアントが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. Memcached

   NAT インスタンスに iptables ルールを追加します。

   NAT インスタンスからクラスターノードにキャッシュポートを転送するには、クラスター内の各ノードのNATテーブルに iptables ルールを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 11211 -j DNAT --to
 10.0.1.230:11211
```

   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 Memcached 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 11211 -j DNAT --to
 10.0.1.230:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11212 -j DNAT --to
 10.0.1.231:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11213 -j DNAT --to
 10.0.1.232:11211
```

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。

信頼されたクライアントは、NATインスタンスEIPに関連付けられた に接続し、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。例えば、 の接続文字列は次のPHPようになります。

```
$memcached->connect( '203.0.113.73', 11211 );
$memcached->connect( '203.0.113.73', 11212 );
$memcached->connect( '203.0.113.73', 11213 );
```

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 11211
telnet 203.0.113.73 11212
telnet 203.0.113.73 11213
```

Valkey または Redis OSS

NAT インスタンスに iptables ルールを追加します。

NAT インスタンスからクラスターノードにキャッシュポートを転送するには、クラスター内の各ノードのNATテーブルに iptables ルールを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。例えば、ポート 6379 ~ 6381 を使用して 3 つのノード Redis OSSクラスターを操作する場合、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
```

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。

信頼されたクライアントは、NATインスタンスEIPに関連付けられた に接続し、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。例えば、 の接続文字列は次のPHP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
```

5. 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 で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC
- 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

での接続エンドポイントの検索 ElastiCache

アプリケーションはエンドポイントを使用して ElastiCache 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のアドレスです。

Valkey または Redis で使用するエンドポイント OSS

- スタンドアロンノード では、読み取りオペレーションと書き込みオペレーションの両方にノードのエンドポイントを使用します。

- Valkery、Valkey、Redis OSS (クラスターモードが無効) クラスター では、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。読み取りオペレーションには個々の Node Endpoints を使用します (API/CLI では、これらはリードエンドポイントと呼ばれます) 。

- Valkey または Redis OSS (クラスターモードが有効) クラスター は、クラスターモードが有効になったコマンドをサポートするすべてのオペレーションで、クラスターの設定エンドポイントを使用します。Valkey クラスターまたは Redis クラスター (Redis OSS3.2) OSS 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から読み取ることができます (API/CLI では、これらはリードエンドポイントと呼ばれます) 。

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。

Memcached で使用するエンドポイント

Memcached を使用した ElastiCache サーバーレスキャッシュの場合は、コンソールからクラスターエンドポイントDNSとポートを取得するだけです。

から AWS CLI、 `describe-serverless-caches` コマンドを使用してエンドポイント情報を取得します。

Linux

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

Windows

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

上記のオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
    "ServerlessCaches": [
        {
            "ServerlessCacheName": "serverless-memcached",
            "Description": "test",
            "CreateTime": 1697659642.136,
            "Status": "available",
            "Engine": "memcached",
            "MajorEngineVersion": "1.6",
            "FullEngineVersion": "21",
            "SecurityGroupIds": [
                "sg-083eda453e1e51310"
            ],
            "Endpoint": {
                "Address": "serverless-memcached-01.amazonaws.com",
                "Port":11211
            },
            "ARN": "<the ARN>",
            "SubnetIds": [
                "subnet-0cf759df15bd4dc65",
                "subnet-09e1307e8f1560d17"
            ],
            "SnapshotRetentionLimit": 0,
            "DailySnapshotTime": "03:00"
        }
    ]
}
```

インスタンスベースの Memcached クラスターの場合は、自動検出を使用すると、クラスターの設定エンドポイントを Memcached クライアントの設定に使用できます。つまり、自動検出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。

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。

Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)

Valkery、Valkey、Redis OSS (クラスターモードが無効) クラスターにノードが 1 つしかない場合、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。Valkery または Valkey または Redis OSS (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、エンドポイントには、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 種類があります。

プライマリエンドポイントは、クラスター内のプライマリノードに常に解決されるDNS名前です。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、 ElastiCache (Redis OSS) 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受信接続を均等に分割します。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 (Redis OSS) 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、レプリカノードの 1 つの IP アドレスにラウンドロビン方式で解決されるDNSレコードです。

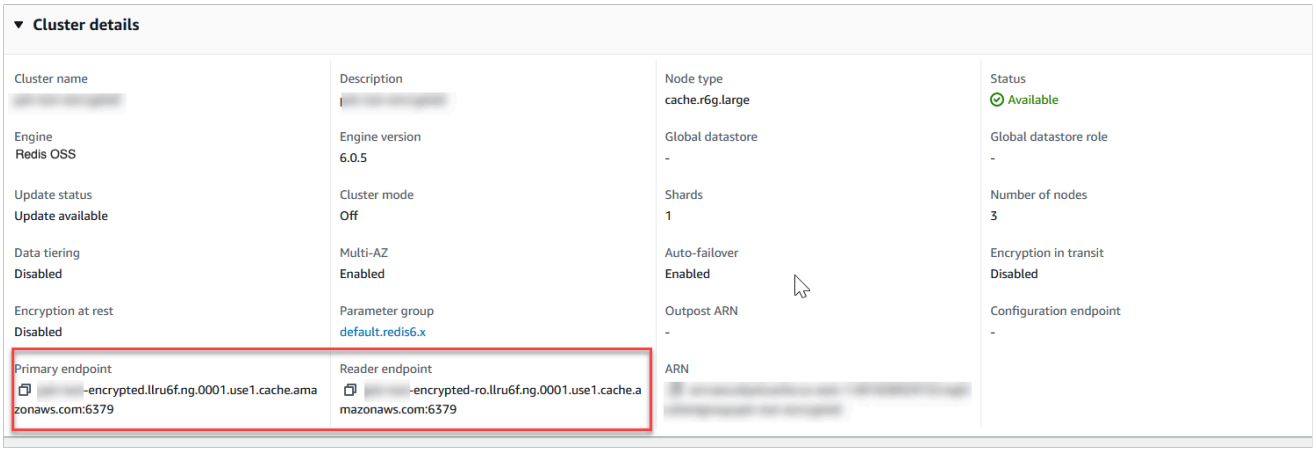
読み取りアクティビティの場合、アプリケーションはクラスター内のいずれのノードにも接続できます。プライマリエンドポイントとは異なり、ノードエンドポイントは特定のエンドポイントに解決されます。レプリカの追加または削除など、クラスターに変更を加えた場合は、アプリケーションでノードエンドポイントを更新する必要があります。

Valkery または Valkey または Redis OSS (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. 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から、Valkey クラスターまたは Redis OSSクラスター を選択します。

クラスター画面には、Valkery または Valkey または Redis OSS (クラスターモードが無効) クラスターと Valkey または Redis OSS (クラスターモードが有効) 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。

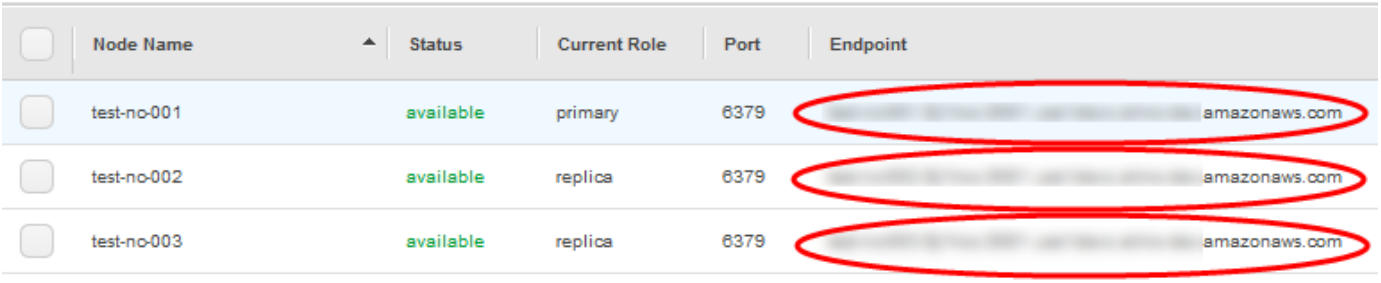


Valkery、Valkey、Redis OSS (クラスターモードが無効) クラスターのプライマリエンドポイントとリーダーエンドポイント

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. Valkery または Valkey または Redis OSS (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、クラスター名を選択し、Nodes タブを選択することで、クラスターのレプリカノードエンドポイントを見つけることができます。

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。



Valkery、Valkey、Redis OSS (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:

   a. 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。
   b. 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、「Memcached ノードへの接続」を参照してください。

Valkery または Valkey または Redis OSS (クラスターモードが無効) のプライマリエンドポイントは次のようになります。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
```

Valkey または Redis OSS (クラスターモードが有効) クラスターのエンドポイントの検出 (コンソール)

Valkey または Redis OSS (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。

Valkey または Redis OSS (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. 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から、Valkey クラスターまたは Redis OSSクラスター を選択します。

   クラスターの一覧が表示されています。接続するクラスターを選択します。

3. クラスターの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。

クラスターのエンドポイントの検索 (コンソール) (Memcached)

すべての Memcached エンドポイントは読み取り/書き込みエンドポイントです。Memcached クラスター内のノードに接続するには、アプリケーションは、各ノードのエンドポイントを使用できるか、クラスターの設定エンドポイントと自動検出を使用できる必要があります。自動検出を使用するには、自動検出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。

自動検出を使用するとき、クライアントアプリケーションは設定エンドポイントを使用して Memcached クラスターに接続します。ノードの追加や削除を行ってクラスターをスケーリングするたびに、アプリケーションは自動的にクラスターのすべてのノードを「検出して、それらのどのノードにも接続できます。アプリケーションで自動検出が行われない場合は、ノードを追加したり削除したりするたびにエンドポイントを手動で更新する必要があります。

エンドポイントをコピーするには、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。エンドポイントを使用してノードに接続する方法については、「Memcached ノードへの接続」を参照してください。

設定エンドポイントとノードエンドポイントはよく似ています。その違いは以下の太字の部分です。

```
myclustername.xxxxxx.cfg.usw2.cache.amazonaws.com:port   # configuration endpoint
 contains "cfg"
myclustername.xxxxxx.0001.usw2.cache.amazonaws.com:port  # node endpoint for node 0001
```

> **Important**
>
> Memcached 設定エンドポイントCNAMEの を作成する場合、自動検出クライアントが を設定エンドポイントCNAMEとして認識するには、 .cfg.に を含める必要がありますCNAME。

エンドポイントの検索 (AWS CLI)

Memcached では、 AWS CLI for Amazon ElastiCache を使用してノードとクラスターのエンドポイントを検出できます。

Redis では、 AWS CLI for Amazon を使用して ElastiCache 、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出できます。

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)
- Valkey または Redis OSS 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)

を使用して、 `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 AWS CLI を検出できます。Valkey または Redis OSSクラスターの場合、 コマンドはクラスターエンドポイントを返します。 Memcached クラスターでは、コマンドは設定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、Memcached クラスター mycluster の設定エンドポイント (`ConfigurationEndpoint`) と個々のノードエンドポイント (`Endpoint`) を取得します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

Windows の場合:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

上記のオペレーションからの出力は、次のようになります (JSON 形式) 。

```
{
```

```
    "CacheClusters": [
    {
        "Engine": "memcached",
        "CacheNodes": [
           {
              "CacheNodeId": "0001",
              "Endpoint": {
                 "Port": 11211,
                 "Address": "mycluster.amazonaws.com"
              },
                 "CacheNodeStatus": "available",
                 "ParameterGroupStatus": "in-sync",
                 "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                 "CustomerAvailabilityZone": "us-west-2b"
           },
           {
              "CacheNodeId": "0002",
              "Endpoint": {
                 "Port": 11211,
                 "Address": "mycluster.amazonaws.com"
              },
                 "CacheNodeStatus": "available",
                 "ParameterGroupStatus": "in-sync",
                 "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                 "CustomerAvailabilityZone": "us-west-2b"
           },
           {
                 "CacheNodeId": "0003",
                 "Endpoint": {
                    "Port": 11211,
                    "Address": "mycluster.amazonaws.com"
                 },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                    "CustomerAvailabilityZone": "us-west-2b"
           }
        ],
        "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "CacheParameterGroupName": "default.memcached1.4",
        "ParameterApplyStatus": "in-sync"
             },
             "CacheClusterId": "mycluster",
```

```
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycluster.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-09-22T21:30:29.967Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "mon:09:00-mon:10:00",
            "CacheNodeType": "cache.m4.large",
             "DataTiering": "disabled"
        }
    ]
}
```

Important

Memcached 設定エンドポイントCNAMEの を作成する場合、自動検出クライアントが を設定エンドポイントCNAMEとして認識するには、 `.cfg.` に を含める必要がありますCNAME。例えば、php.ini ファイルの `mycluster.cfg.local` パラメータで `session.save_path` を含めます。

Example

Valkey と Redis の場合OSS、次のコマンドは単一ノードクラスターの mycluster のクラスター情報を取得します。

Important

パラメータは`--cache-cluster-id`、レプリケーショングループの単一ノードの Valkey または Redis OSS (クラスターモードが無効) クラスター ID または特定のノード ID で使用できます。`--cache-cluster-id` レプリケーショングループの は、 などの 4 桁の値で

す0001。--cache-cluster-id がレプリケーショングループのクラスター (ノード) の ID である場合、 replication-group-idは出力に含まれます。

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、次のようになります (JSON 形式) 。

```
{
    "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、「」トピックを参照してくださいdescribe-cache-clusters。

Valkey または Redis OSS 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)

を使用して 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、次のようになります (JSON 形式) 。

```
{
   "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describe-replication-groups」の「」を参照してください。

エンドポイントの検出 (ElastiCache API)

Memcached では、Amazon ElastiCache API を使用してノードとクラスターのエンドポイントを検出できます。

Redis では、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出できます。

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)
- Valkey または Redis OSS 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)

を使用して ElastiCache API、 `DescribeCacheClusters`アクション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。Valkey または Redis OSSクラスターの場合、 コマンドはクラスターエンドポイントを返します。 Memcached クラスターでは、コマンドは設定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

Memcached の場合、次のコマンドは Memcached クラスター mycluster の設定エンドポイント (`ConfigurationEndpoint`) と個々のノードエンドポイント (`Endpoint`) を取得します。

```
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>
```

> **Important**
>
> Memcached 設定エンドポイントCNAMEの を作成する場合、自動検出クライアントが を設定エンドポイントCNAMEとして認識するには、 `.cfg.`に を含める必要があ

りますCNAME。例えば、php.ini ファイルの mycluster.*cfg*.local パラメータで session.save_path を含めます。

Valkey または Redis OSS 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)

を使用して 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。

次のオペレーションは、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 (PrimaryEndpoint) 、リーダーエンドポイント (ReaderEndpoint) 、および個々のノードエンドポイント (ReadEndpoint) を取得しますmyreplgroup。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、「」を参照してくださいDescribeReplicationGroups。

でのシャードの使用 ElastiCache

シャード (API/CLI: ノードグループ) は、Valkey または Redis OSSノード ElastiCache を持つ 1 ~ 6 のコレクションです。Valkey クラスターまたは Redis OSS (クラスターモードが無効) クラスターに複数のシャードが発生することはありません。シャードを使用すると、大き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、速く、より簡単に管理できる部分に分割できます。これにより、複数の別々のセクションにオペレーションを分散することで、データベースの効率を高めることができます。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、多くの利点が得られます。

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。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。

エンジンバージョン Valkey 7.2 または Redis 5.0.6 以降では、ノードまたはシャードの制限をクラスターごとに最大 OSS 500 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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることがあります。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。

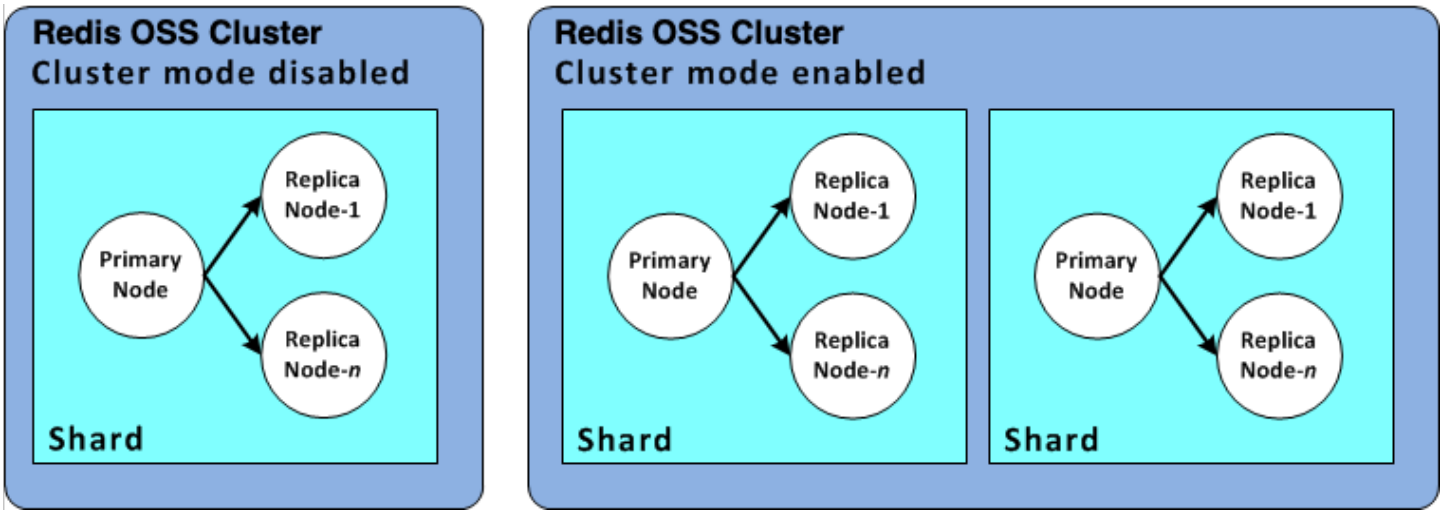
5.0.6 未満のバージョンの場合、クラスターあたりの制限は 250 個です。

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 Valkey または Redis OSS (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の数とシャード内のノードの数を指定します。詳細については、「Valkey または Redis OSS (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。または を使用して ElastiCache APIクラスター AWS CLI を作成する場合 (API/ではレプリケーショングループと呼ばれますCLI) 、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のシャード全体の属性を制御できます。



Valkey または Redis OSS シャード設定

詳細については、「Valkey または Redis のオフライン再シャーディング OSS (クラスターモードが有効)」および「Valkey または Redis のオンライン再シャーディング OSS (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 を使用して見つけることができます ElastiCache API。

の使用 AWS Management Console

トピック

- Valkey または Redis の場合 OSS (クラスターモードが無効)
- Valkey または Redis の場合 OSS (クラスターモードが有効)

Valkey または Redis の場合 OSS (クラスターモードが無効)

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループのシャードは常に IDsです0001。

Valkey または Redis の場合 OSS (クラスターモードが有効)

次の手順では AWS Management Console 、 を使用して、Valkey または Redis OSS (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。

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループでシャード ID 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 ElastiCache コンソールを開きます。
2. ナビゲーションペインで、Valkey または Redis OSSを選択し、シャードを検索する Valkey または Redis OSS (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IDs。
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

Valkey 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、または Valkey または Redis OSS (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 OSS (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を使用して 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 です。*highlighted* 見つけやすくするために、こちらで検索します。

```
{
    "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": {}
        }
    ]
}
```

の使用 ElastiCache API

Valkey 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、または Valkey または Redis OSS (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 OSS (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を使用して 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した場合、最大 の詳細 *xxx* レプリケーショングループが返されます。

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>
```

Valkey、RedisOSS、および Memcached のセルフデザインキャッシュの比較

Amazon は、Valkey、RedisOSS、および Memcached 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、独自設計のクラスター、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、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。

以下が当てはまる ElastiCache 場合は、 OSSで Valkey または Redis を選択します。

- ElastiCache Valkey 7.2 または Redis OSSバージョン 7.0 を使用 (拡張)

  Functions 、Shaded Pub/Sub 、またはACL改善 を使用します。詳細については、「Redis OSS バージョン 7.0 (拡張)」を参照してください。
- ElastiCache (Redis OSS) バージョン 6.2 (拡張)

メモリと r6gd ノードタイプSSDを使用してデータを階層化できます。詳細については、データ階層化を参照してください。

- ElastiCache (Redis OSS) 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 OSSバージョン 6.0 (拡張)」を参照してください。

- ElastiCache (Redis OSS) バージョン 5.0.0 (拡張)

  プロデューサーが新しい項目をリアルタイムで追加し、消費者がメッセージをブロックまたは非ブロックの方法で消費できるようにするログデータ構造である Redis OSSストリーム を使用します。

  詳細については、「Redis OSSバージョン 5.0.0 (拡張)」を参照してください。

- ElastiCache (Redis OSS) バージョン 4.0.10 (拡張)

  Valkey または Redis OSS (クラスターモードが有効) クラスターの暗号化とシャードの動的追加または削除の両方をサポートします。

  詳細については、「Redis OSS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 (Redis OSS) バージョン 3.2.10 (拡張)

  Valkey または Redis OSS (クラスターモードが有効) クラスターでシャードを動的に追加または削除できます。

  > **Important**
  >
  > 現在 ElastiCache (Redis OSS) 3.2.10 は暗号化をサポートしていません。

  詳細については、次を参照してください。

  - Redis OSSバージョン 3.2.10 (拡張)
  - Redis のオンライン再シャーディングのベストプラクティスOSS、詳細については、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ディング
    - Valkey または Redis のオンラインリシャーディングとシャードリバランシング OSS (クラスターモードが有効)
  - Redis OSSクラスターのスケーリングの詳細については、「スケーリング」を参照してください。

- ElastiCache (Redis OSS) バージョン 3.2.6 (拡張)

  以前の Redis OSSバージョンと以下の機能が必要な場合は、 ElastiCache (Redis OSS) 3.2.6 を選択します。

  - 転送時の暗号化 詳細については、「Amazon ElastiCache (Redis OSS) 転送中の暗号化」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 (Redis OSS) At-Rest Encryption 」を参照してください。
- ElastiCache (Redis OSS) (クラスターモードが有効) バージョン 3.2.4

  Redis 2.8.x OSS の機能に加えて以下の機能が必要な場合は、Redis 3.2.4 (クラスターモード) OSS 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 (Redis OSS) (非クラスターモード) 2.8.x および 3.2.4 (拡張)

  以下に該当する場合は、Redis 2.8.x OSS または Redis 3.2.4 (クラスター化されていないモード) OSS 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。

- サーバーレスキャッシュだけでなく、独自設計のクラスターのバックアップおよび復元機能も必要です。
- 複数のデータベースをサポートする必要がある。

Memcached、Valkey または Redis OSS (クラスターモードが無効)、および Valkey または Redis OSS (クラスターモードが有効) の比較の概要

	Memcached	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (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	可能	いいえ	可能
クラスターが変更可能	あり	可能	3.2.10 以降 - 限定
オンラインリシャーディング	不可	なし	3.2.10 以降
暗号化	転送中 1.6.12 以降	4.0.10 以降	4.0.10 以降
データ階層化	不可	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			
連邦RAMP	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	可能	4.0.10 以降	4.0.10 以降
マルチスレッド	可能	いいえ	なし
ノードタイプのアップグレード	不可	はい	可能

	Memcached	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (クラスターモードが有効)
エンジンのアップグレード	あり	はい	可能
高可用性 (レプリケーション)	不可	はい	可能
自動フェイルオーバー	不可	オプションです。	必須
パブリック/サブ機能	不可	はい	可能
ソートされたセット	不可	はい	可能
バックアップと復元	Serverless Memcached 専用。独自設計の Memcached クラスター用ではありません。	あり	可能
地理空間インデックス作成	不可	4.0.10 以降	可能

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

+ 廃止されたバージョン、すでに有効期限切れになったバージョン、または間もなく有効期限切れになるバージョンは除外されます。

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、「サポートされているノードの種類」を参照してください。

Valkey または Redis のオンライン移行 OSS

オンライン移行を使用すると、Amazon OSSのセルフホスト型のオープンソース Valkey または Redis から Amazon EC2にデータを移行できます ElastiCache。

> **Note**
>
> オンライン移行は、r6gd ノードタイプで実行されている ElastiCache サーバーレスキャッシュまたはクラスターではサポートされていません。

概要

Amazon でOSS実行されているオープンソースの Valkey または Redis から Amazon EC2にデータを移行するには、既存の Amazon デプロイまたは新しく作成された Amazon ElastiCache デプロイ ElastiCache 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 OSSのセルフホスト型のオープンソース Valkey または Redis から EC2 へのデータ移行用に設計されており ElastiCache、 ElastiCache クラスター間でデータを移動する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。

移行は、 `StartMigration` API オペレーションまたは AWS CLI コマンドを呼び出すときに開始されます。Valkey または Redis OSSクラスターモードが無効になっているクラスターを移行する場合、移行プロセスにより、 ElastiCache Valkey または Redis OSSクラスターのプライマリノードがソース Valkey または Redis OSSプライマリのレプリカになります。Valkey または Redis OSSクラスターモードが有効になっているクラスターを移行する場合、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ー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、 CompleteMigrationAPIオペレーションを呼び出します。このAPIオペレーションは、プライマリノードとレプリカノード (該当する場合) を使用して、プライマリ Valkey または Redis デプロイ ElastiCache にOSSデプロイを促進します。これで、クライアントアプリケーションをリダイレクトして、 へのデータの書き込みを開始できるようになりました ElastiCache。移行中は、Valkey ノードとプライマリノードで valkey-cli INFO コマンドを実行して、レプリケーションのステータスを確認できます。 ElastiCache

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。

- 移行のためのソースとターゲットの準備
- 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

移行のためのソースとターゲットの準備

これらのステップでは、セルフホストの Valkey または Redis ソースから EC2 に ElastiCache、または Redis OSSクラスターから ElastiCache Valkey クラスターにデータを移行するように準備できます。

ElastiCache コンソールAPIまたは から移行を開始する前に、以下に記載されている 4 つの前提条件がすべて満たされていることを確認する必要があります AWS CLI。

移行のためにソースノードとターゲットの Valkey ノードまたは Redis OSSノードを準備するには

1. ターゲット ElastiCache デプロイを特定し、そのデプロイにデータを移行できることを確認します。

   既存のデプロイまたは新しく作成された ElastiCache デプロイは、移行に関する以下の要件を満たしている必要があります。

   - Valkey または Redis OSS エンジンバージョン 5.0.6 以降を使用しています。

- 転送中の暗号化が有効になっていません。
- マルチ AZ が有効になっている。
- Valkey または Redis OSS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ります。予約メモリを適切に設定するには、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。
- クラスターモードが無効になっている場合、 または CLIコンソールを使用して、 または Valkey または Redis OSSバージョン OSS 5.0.6 以降を使用している場合、Valkey または Redis バージョン 2.8.21 以降から CLI Valkey または Redis OSS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になっている場合、 または CLIコンソールを使用して または Redis OSSバージョン OSS5.0.6 以降を使用している場合、クラスターモードが有効になっている任意の Valkey CLIまたは Redis バージョンから Redis OSS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 Valkey または Redis OSSと ElastiCache デプロイの設定に互換性があることを確認してください。

   少なくとも、ターゲット ElastiCache デプロイの以下のすべてが、レプリケーション用の Valkey または Redis OSS設定と互換性がある必要があります。

   - クラスターAUTHを有効にしないでください。
   - 設定は に設定`protected-mode`する必要がありますno。
   - Valkey または Redis `bind`の設定がある場合はOSS、 ElastiCache ノードから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 Valkey または Redis OSSクラスターで同じである必要があります。この値は、Valkey または Redis 設定`databases`で OSS を使用して設定されます。
   - データ変更を実行する Valkey または Redis OSS コマンドの名前を変更して、データのレプリケーションが成功するようにしないでください。例えば`sync`、`psync`、、`info`、`config`、`command`、、および ですcluster。
   - Valkey または Redis OSSクラスターから にデータをレプリケートするには ElastiCache、この追加ロードを処理するのに十分なCPUメモリとメモリがあることを確認してください。このロードは、Valkey または Redis OSSクラスターによって作成され、ネットワーク経由で ElastiCache ノードに転送されたRDBファイルから取得されます。

- ソースクラスター内のすべての Valkey インスタンスまたは Redis OSS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 ElastiCache 、インスタンスが 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 Valkey または Redis と同じ仮想プライベートクラウド (VPC) OSSに ElastiCache デプロイを割り当てまたは作成します (推奨) 。
   - VPCs が異なる場合は、ノード間のアクセスを許可するピアVPCリングを設定します。VPC ピアリングの詳細については、「」を参照してくださいAmazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC。
   - Valkey インスタンスまたは Redis OSS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ら、アプリケーションが ElastiCache ノードにトラフィックを誘導できることを確認してください。詳細については、「Amazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC」を参照してください。

データ移行のテスト

すべての前提条件が完了したら、 AWS Management Console、 ElastiCache API、または を使用して移行設定を検証できます AWS CLI。次の例は、 の使用を示していますCLI。

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。

以下は、 を使用する例ですCLI。

```
aws elasticache test-migration --replication-group-id test-cluster --customer-node-
endpoint-list "Address='10.0.0.241',Port=6379"
```

ElastiCache は、実際のデータ移行なしで移行設定を検証します。

移行の開始

すべての前提条件が完了したら、 AWS Management Console、 ElastiCache API、または を使用してデータ移行を開始できます AWS CLI。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。次の例は、 の使用を示していますCLI。

> **Note**
>
> TestMigration API を使用して移行設定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 – DNSまたは IP アドレスのいずれかと、ソース Valkey または Redis OSSクラスターが実行されているポートを持つエンドポイントのリスト。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 Valkey または Redis OSSクラスターのプライマリノードではなくレプリカを指すことができます。

以下は、 を使用する例ですCLI。

```
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 (各シャード内) は Valkey インスタンスまたは Redis OSSインスタンス (クラスターで有効な redis 内の同じスロットを所有する対応するシャード内) のレプリカになるように自身を設定します。 ElastiCache クラスターのステータスが移行に変わり、データが Valkey または Redis OSSインスタンスから ElastiCache プライマリノードへの移行を開始します。Valkey または Redis OSSインスタンスのデータのサイズとロードによっては、移行が完了するまでに時間がかかる場合があります。Valkey インスタンスとプライマリノードで valkey-cli INFO コマンドを実行して、移行の進行状況を確認できます。 ElastiCache

レプリケーションが成功すると、Valkey または Redis OSSインスタンスへのすべての書き込みはクラスターに伝達されます ElastiCache。 ElastiCache ノードは読み取りに使用できます。ただし、ク

ラスターに ElastiCache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引き続き ElastiCache プライマリノードからレプリケートされます。これにより、Valkey または Redis OSSクラスターのすべてのデータが ElastiCache クラスター内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 Valkey インスタンスまたは Redis OSSインスタンスのレプリカになれない場合は、数回再試行してから、最終的にプライマリに昇格します。その後、クラスターのステータス ElastiCacheが使用可能な に変わり、移行の開始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、「移行のためのソースとターゲットの準備」のガイドラインに従っていることを確認します。
- VPC および サブネットのルーティング設定で、 ElastiCache ノードと Valkey または Redis OSS インスタンス間のトラフィックが許可されていることを確認します。
- Valkey インスタンスまたは Redis OSS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 ノードからの入力バインドトラフィックを許可していることを確認します。
- レプリケーションに固有の障害の詳細については、インスタンスの Valkey または Redis OSS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。

- Valkey または Redis OSS`master_link_status`が ElastiCache プライマリノード (複数可) の `INFO` コマンドupにあることを確認します。この情報はコンソールでも確認できます ElastiCache 。クラスターを選択し、CloudWatch メトリクス で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。
- Valkey インスタンスまたは Redis OSSインスタンスで `INFO` コマンドを実行すると、ElastiCache レプリカにオンライン状態があることを確認できます。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。
- Valkey または Redis OSSインスタンスの CLIENT LIST コマンドを使用して、クライアント出力バッファが低いことを確認します。

データ移行が完了すると、データは Valkey または Redis OSSクラスターのプライマリノード (複数可) に送信される新しい書き込みと同期されます。

データ移行の完了

ElastiCache クラスターにカットオーバーする準備ができたら、次のパラメータで `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 (各シャード内) は Valkey または Redis OSS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 書き込みと読み取りを指示できます。 ElastiCache クラスターのステータスは、 から使用可能な への移行に変わります。

プライマリへの昇格が失敗しても、 ElastiCache プライマリノードは Valkey または Redis OSSインスタンスからレプリケートし続けます。 ElastiCache クラスターは引き続き移行ステータスになり、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表示されます。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 Valkey インスタンスまたは Redis OSS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 つが置き換えられている場合、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

を使用して AWS Management Console 、クラスターから Valkey または Redis OSS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、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. 新しい Valkey または Redis OSS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。
   - エンジンバージョンは Valkey 7.2 以降、または Redis 5.0.6 OSS 以降である必要があります。
   - クラスターAUTHを有効にしないでください。
   - 設定は に設定`protected-mode`する必要がありますno。
   - Valkey または Redis `bind`の設定がある場合はOSS、 ElastiCache ノードから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 Valkey または Redis OSSクラスター間で同じである必要があります。この値は、エンジン設定`databases`で を使用して設定されます。
   - データ変更を実行する Valkey または Redis OSS コマンドの名前を変更して、データのレプリケーションが成功するようにしないでください。
   - Valkey または Redis OSSクラスターから にデータをレプリケートするには ElastiCache、この追加ロードを処理するのに十分なCPUメモリとメモリがあることを確認してください。このロードは、Valkey または Redis OSSクラスターによって作成され、ネットワーク経由で ElastiCache ノードに転送されたRDBファイルから取得されます。
   - 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に、IP アドレスと、Valkey または Redis OSSクラスターが利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`*`.`

のリージョンとアベイラビリティーゾーンの選択 ElastiCache

対応するエンドポイントを使用してリージョンとアベイラビリティーゾーンを指定することで、ElastiCache クラスターのスケーラビリティと信頼性を高めることができます。

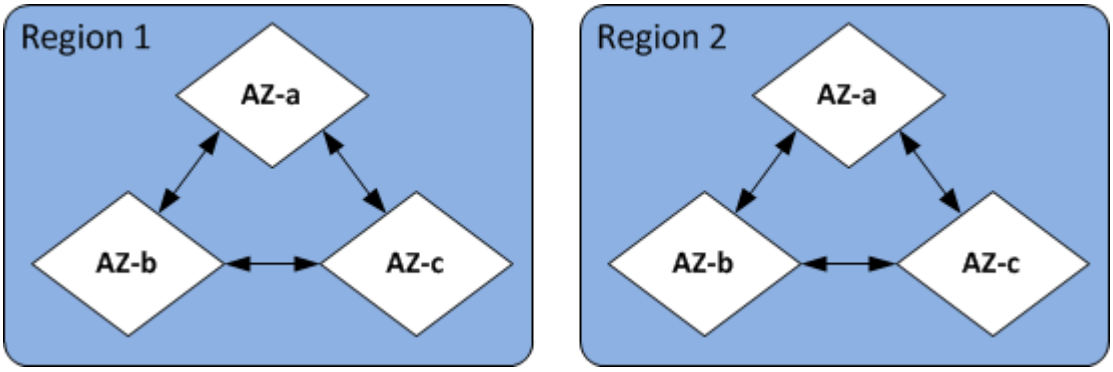
AWS クラウドコンピューティングリソースは、可用性の高いデータセンター施設に格納されています。スケーラビリティと信頼性を向上させるために、これらのデータセンターの設備は物理的に異なる場所に配置されています。これらの場所は、リージョンとアベイラビリティーゾーンに分類されます。

AWS リージョンは大きく、地理的に別の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、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た AWS リージョン内の個

別の場所です。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安価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

- Memcached でのアベイラビリティーゾーンに関する考慮事項
- 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
- でのローカルゾーンの使用 ElastiCache
- での Outposts の使用 ElastiCache

Memcached でのアベイラビリティーゾーンに関する考慮事項

リージョン内の複数のアベイラビリティーゾーンに Memcached ノードを分散することで、アベイラビリティーゾーン内の停電など、壊滅的な障害の影響から保護できます。

サーバーレスキャッシュ

ElastiCache サーバーレスキャッシュは、複数のアベイラビリティーゾーンにまたがる高可用性キャッシュを作成します。異なるアベイラビリティーゾーンからサブネットを指定することも、サーバーレスクラスターを作成するVPCのと同じ方法で指定することもElastiCache 、デフォルトの からサブネットを自動的に選択することもできますVPC。

独自の ElastiCache (Memcached) クラスターの設計

Memcached クラスターでは、300 個までのノードを設定できます。Memcached クラスターにノードを作成または追加するときは、すべてのノードに単一のアベイラビリティーゾーンを指定したり、すべてのノード ElastiCache に単一のアベイラビリティーゾーンを選択したり、各ノードにアベイラビリティーゾーンを指定したり、各ノードにアベイラビリティーゾーン ElastiCache を選択したりできます。新しいノードは、既存の Memcached クラスターに追加するときに、異なるアベイラビリティーゾーンで作成できます。キャッシュノードが作成されると、そのアベイラビリティーゾーンは変更できません。

単一のアベイラビリティーゾーンクラスター内のクラスターを複数のアベイラビリティーゾーンに分散させる場合は、さまざまなアベイラビリティーゾーンに新しいノード ElastiCache を作成できます。その後、元のキャッシュノードの一部またはすべてを削除できます。この手法をお勧めします。

Memcached ノードを単一のアベイラビリティーゾーンから複数のアベイラビリティーゾーンに移行するには

1. 必要なアベイラビリティーゾーンに新しいキャッシュノードを作成して、クラスターを変更します。リクエストで、以下の操作を実行します。

   - AZMode (CLI: `- -az-mode`) を に設定しますcross-az。
   - NumCacheNodes (CLI: `- -num-cache-nodes`) を、現在アクティブなキャッシュノードの数と、作成する新しいキャッシュノードの数に設定します。
   - （ NewAvailabilityZonesCLI: `- -new-availability-zones`) を、新しいキャッシュノードを作成するゾーンのリストに設定します。新しいノードごとにアベイラビリティーゾーンElastiCache を決定できるようにするには、リストを指定しないでください。
   - ApplyImmediately (CLI: `- -apply-immediately`) を true に設定します。

   > **Note**
   >
   > 自動検出を使用していない場合は、必ず新しいキャッシュノードエンドポイントでクライアントアプリケーションを更新してください。

次の手順に進む前に、Memcached ノードが完全に作成され、使用可能であることを確認してください。

2. 元のアベイラビリティーゾーンで不要になったノードを削除して、クラスターを変更します。リクエストで、以下の操作を実行します。

   - この変更が適用された後に必要なアクティブなキャッシュノードの数に NumCacheNodes (CLI: - -num-cache-nodes) を設定します。
   - CacheNodeIdsToRemove (CLI: - -nodes-to-remove) を、クラスターから削除するキャッシュノードのリストに設定します。

     IDs リストされているキャッシュノードの数は、現在アクティブなノードの数から の値を引いた数に等しくなければなりませんNumCacheNodes。
   - （ オプション) ApplyImmediately (CLI: - -apply-immediately) を true に設定します。

     ApplyImmediately (CLI: - -apply-immediately) を true に設定しない場合、ノードの削除は次のメンテナンスウィンドウで行われます。

ノードの配置

Amazon ElastiCache 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 () に配置することをサポートしていますAZs。さらに、ノードを複数の AZs (推奨) に配置する場合、ノードごとに AZ を選択する ElastiCache か、 ElastiCache でノードを選択することができます。

別の にノードを配置することでAZs、1 つの AZ で停電などの障害が発生すると、システム全体が失敗する可能性がなくなります。テストでは、1 つの AZ 内のすべてのノードを見つけるか、複数の にまたがってノードを分散させることで、レイテンシーに大きな差がないことが示されていますAZs。

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。

- Memcached 用のクラスターの作成
- Valkey または Redis 用のクラスターの作成 OSS
- ElastiCache クラスターの変更
- ElastiCache 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us-west-1、高可用性のために複数のアベイラビリティーゾーン (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、耐障害性AZsを実現するためにノードを別の で起動することを選択できます。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「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」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(バージニア州北部) リージョン us-east-1	elasticache.us-east-1.amazonaws.com	HTTPS	
US West (N. California) Region 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 (UAE) 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を参照してください。ElastiCache			

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

でのローカルゾーンの使用 ElastiCache

ローカルゾーンは、地理的にユーザーに近い AWS リージョンの拡張です。新しいサブネットを作成してローカルゾーンに割り当てることで、仮想プライベートクラウド (VPC) を親 AWS リージョンからローカルゾーンに拡張できます。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、 の他のサブネットと同じように動作しますVPC。

ローカルゾーンを使用すると、 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は、ローカルゾーンで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。

ローカルゾーンはリージョン AWS コードで表され、その後に などの場所を示す識別子が続きます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。

for ElastiCache Local Zones 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「ローカルゾーンの有効化」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「 でのサブネットの作成VPC」を参照してください。

3. ローカルゾーンに 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は、ローカルゾーンのアベイラビリティーゾーングループを選択します。

   詳細については、「サブネットグループの作成」を参照してください。

4. ローカルゾーンで ElastiCache サブネットを使用する ElastiCache (Memcached) クラスターを作成します。

   詳細については、「Memcached クラスター (CLI) の作成 (コンソール)」を参照してください。

5. ローカルゾーンで ElastiCache サブネットを使用する ElastiCache (Redis OSS) 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。

   - Valkey (クラスターモードが無効) クラスターの作成 (コンソール)
   - Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)

での Outposts の使用 ElastiCache

AWS Outposts は で使用できます ElastiCache。Outposts は、 AWS インフラストラクチャ、サービス、APIs、およびツールをお客様の施設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、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とローカルデータ処理のニーズを低く抑えながら、 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できるようにします。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、 AWS リージョンの一部としてこの容量を AWS 運用、モニタリング、管理します。Outpost で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サブネットを指定できます。

> **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 on Outposts は CoIP 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。

ElastiCache 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、Valkey キャッシュ 、Redis OSSキャッシュ 、または Memcached キャッシュ のいずれかを選択します。
3. Valkey キャッシュ を選択した場合は、Valkey キャッシュの作成 を選択します。Redis OSS キャッシュ を選択した場合は、Redis OSSキャッシュの作成 を選択します。Memcached キャッシュ を選択した場合は、Create Memcached cache 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。クラスターモードを無効 に設定したままにします。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. 場所については、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は、Outpost ID 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. Connectivity で、新しいサブネットグループの作成 を選択し、VPCID を入力します。残りを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. Valkey 設定 、Redis OSS設定 、または Memcached 設定 で、使用するエンジンに応じて、次のように設定します。

a. 名前 : クラスターの名前を入力します。

b. 説明 : 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 AWS Outpost リージョンに基づいています

d. ポート : Valkey または Redis の場合はOSS、デフォルトのポート 6379 を受け入れます。Memcached の場合、デフォルトのポート 11211 を受け入れます。別のポートを使用する場合は、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。Valkey または Redis を使用している場合OSS、Porting Assistant for 。NET 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。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。リードレプリカの数は少なくとも 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 と一致する必要がありますVPC。Outposts IDsにサブネットVPCがない を選択すると、リストは空を返します。
   - [アベイラビリティーゾーンまたは 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。IDs 使用可能なサブネットがない場合は、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

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。

- アベイラビリティーゾーン : 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。

での Outposts の使用 AWS CLI

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、スクリプトを使用して自動化できます。は AWS CLI、アドホック (1 回限り) 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または Linux 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。

をダウンロード、インストール、設定するには CLI

1. AWS コマンドラインインターフェイスウェブページで 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「 のインストール AWS CLI」および「 の設定 AWS CLI」の手順に従います。

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すると、 `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 で作成されるか、複数の Outpost で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の詳細を確認できます。

トピック

スナップショットおよび復元

Valkey、Redis OSS、または Serverless Memcached 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。新しい Valkey、Redis、または Serverless Memcached キャッシュを作成しOSS、バックアップからのデータを入力して、いつでもデータを復元できます。を使用すると ElastiCache、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、および を使用してバックアップを管理できます ElastiCache API。

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

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、Valkey、Redis、OSS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。
- Valkey または Redis OSS (クラスターモードが無効) クラスターでは、バックアップと復元は`cache.t1.micro`ノードではサポートされていません。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。
- Valkey または Redis OSS (クラスターモードが有効) クラスターでは、バックアップと復元はすべてのノードタイプでサポートされています。
- 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。Valkey および Redis のOSS独自設計クラスターでは、クラスター内のノードごとに 20 個以下の手動バックアップを作成できます。
- Valkey または Redis OSS (クラスターモードが有効) は、クラスターレベル (APIまたは の場合CLI、レプリケーショングループレベル) でのバックアップのみをサポートします。Valkey または Redis OSS (クラスターモードが有効) は、シャードレベル (APIまたは の場合CLI、ノードグループレベル) でのバックアップの取得をサポートしていません。

- バックアッププロセス中、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、独自設計のクラスターで 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 Valkey または Redis OSSキャッシュを使用している場合、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。独自設計クラスターのバックアップは ElastiCache (Memcached) では利用できませんが、 ElastiCache (Redis) では利用できますOSS。

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には、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、Valkey と Redis がノードのすべての使用可能なメモリを消費OSSするのを防ぎ、ページングの量を減らすのに役立ちます。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。予約メモリとreserved-memory-percentパラメータの詳細については、「」を参照してくださいValkey と Redis の予約済みメモリの管理 OSS。

- リードレプリカからバックアップを作成する – 複数のノードを持つノードグループOSSで Valkey または Redis を実行している場合は、プライマリノードまたはリードレプリカの 1 つからバックアップを取ることができます。中に必要なシステムリソースがあるためBGSAVE、リードレプリカの 1 つ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 Valkey または Redis OSS データを確実にキャプチャできます。

自動バックアップのスケジュール

任意の Valkey または Redis OSSサーバーレスキャッシュまたは独自設計クラスターの自動バックアップを有効にすることができます。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。

Memcached Serverless キャッシュの自動バックアップを有効にすることができます。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:

- バックアップの開始時刻 – 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、 はバックアップウィンドウを自動的に 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。

自動バックアップをスケジュールすると、 ElastiCache はバックアップの作成を開始します。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、 はバックアップウィンドウを自動的に ElastiCache 割り当てます。

ElastiCache コンソール、、または を使用して、新しいキャッシュの作成時または既存のキャッシュの更新時に自動バックアップを有効 AWS CLIまたは無効にできます ElastiCache API。Valkey および Redis の場合OSS、これは、アドバンストバルキー設定またはアドバンスト Redis OSS

設定セクションの「自動バックアップを有効にする」ボックスにチェックマークを付けて行います。Memcached の場合、これは、アドバンスト Memcached 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して行われます。

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 を使用して作成できます ElastiCache API。

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、 で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインから、好みに応じて、Valkey キャッシュ 、Redis OSSキャッシュ 、または Memcached 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、次のパラメータで `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
```

を使用した自己設計型クラスターの手動バックアップ AWS CLI

を使用して自己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、次のパラメータで `create-snapshot` AWS CLI オペレーションを使用します。

- `--cache-cluster-id`
  - バックアップしているクラスターにレプリカノードがない場合、 `--cache-cluster-id`はバックアップしているクラスターの名前です。例えば、*`mycluster`*.

- 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。例えば、 の名前は です。*mycluster-002*.

  このパラメータは、Valkey または Redis OSS (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 Valkey または Redis OSS (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。Valkey または Redis OSS (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場合は、このパラメータを使用します。

- `--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 Valkey または Redis OSS (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 Valkey または Redis OSS (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: レプリカノードを使用した Valkey または Redis OSS (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、Valkey または Redis OSS (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
```

出力例: レプリカノードを使用した Valkey または Redis OSS (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: Valkey または Redis のクラスターのバックアップ OSS (クラスターモードが有効)

次の AWS CLI オペレーションでは、Valkey または Redis OSS (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
```

出力例: Valkey または Redis OSS (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

ElastiCache コンソール、、 AWS CLIまたは を使用して、最終バックアップを作成できます ElastiCache API。

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、Valkey または Redis OSSサーバーレスキャッシュ、Valkey または Redis OSSの独自設計クラスター、または Memcached サーバーレスキャッシュを削除すると、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で「バックアップの作成」の「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

- の使用 AWS Management Console
- 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

- Valkey キャッシュ、Redis OSSキャッシュ、または Memcached サーバーレスキャッシュを削除する場合
- リードレプリカのない Valkey または Redis OSSクラスターを削除する場合
- リードレプリカを使用して Valkey または Redis OSSクラスターを削除する場合

Valkey キャッシュ、Redis OSSキャッシュ、または Memcached 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、以下のパラメータで `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delete-serverless-cache」の「」を参照してください。

リードレプリカのない Valkey または Redis OSSクラスターを削除する場合

リードレプリカのない自己設計型クラスターの最終バックアップを作成するには、次のパラメータで 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delete-cache-cluster」の「」を参照してください。

リードレプリカを使用して Valkey または Redis OSS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、次のパラメータで 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、「 コマンドリファレンスdelete-replication-group」の「」を参照してください。 AWS CLI

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、オプションで特定のバックアップに関する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describe-serverless-cache-snapshotsの「」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)

独自設計のクラスターバックアップのリストと、オプションで特定のバックアップに関する詳細を表示するには、 describe-snapshotsCLI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、 AWS CLI コマンドリファレンスの describe-snapshots を参照してください。

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、「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。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 (Redis OSS) 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、 の外部からバケットにアクセスできます ElastiCache。 ElastiCache コンソール、、 AWS CLIまたは を使用してバックアップをエクスポートできます ElastiCache API。

バックアップをエクスポートすると、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キャッシュをシードできます。新しいクラスターが使用中に入力されるのを待つ必要はありません。新しいクラスターのシードについては、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」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、オフライン処理に .rdb ファイルを使用することです。

Important

- ElastiCache バックアップとコピー先の Amazon S3 バケットは、同じ AWS リージョンにある必要があります。

  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、「のデータ階層化 ElastiCache」を参照してください。
- バックアップのエクスポートは、Valkey と Redis OSSの独自設計クラスター、Serverless Valkey と Redis OSS、および Serverless Memcached で使用できます。バックアップのエクスポートは、独自設計の Memcached クラスターでは使用できません。

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。

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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
2. バケットの作成 を選択します。
3. バケットを作成する - バケット名と地域を選択する で、以下の操作を実行します。

   a. バケット名に Amazon S3 バケットの名前を入力します。

      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. [Create] (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。

Amazon S3 バケット ElastiCache へのアクセスを許可する

ElastiCache がスナップショットを Amazon S3 バケットに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ための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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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、「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. アクセス許可タブ と アクセス許可 を選択し、アクセスコントロールリスト (ACL) を選択し、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)
   - バケット ACL: 読み取り、書き込み

   > **Note**
   >
   > - PDT GovCloud リージョンの場合、正規 ID は `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6` です。
   > - OSU GovCloud リージョンの場合、正規 ID は `c54286759d2a83da9c480405349819c993557275cf37d820d514b42da6893f5c` です。

5. [Save] を選択します。

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、それにアクセスする ElastiCache アクセス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。次の例では、発信者の IAM ID に次の追加の S3 固有のIAMアクセス許可があ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、 ElastiCache コンソールを使用して Amazon S3 バケットにバックアップをエクスポートし、 の外部からバックアップにアクセスできます ElastiCache。Amazon S3 バケットは、ElastiCache バックアップと同じ AWS リージョンにある必要があります。

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。

    名前は 1 ~ 1,000 文字で、UTF-8 エンコードできる必要があります。

    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 (「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、次のアクセス許可が必要です。

   - 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、「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。「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WRITE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

次のパラメータを持つ export-serverless-cache-snapshotCLIオペレーションを使用して、バックアップを Amazon S3 バケットに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、 がバックアップの AWS リージョンにある Amazon S3 バケットで、次のアクセス許可を持っている--s3-bucket-name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
```

独自設計の 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持つ copy-snapshotCLIオペレーションを使用して、バックアップを Amazon S3 バケットに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。

名前は 1 ~ 1,000 文字で、UTF-8 エンコードできる必要があります。

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、 がバックアップの AWS リージョンにある Amazon S3 バケットで、次のアクセス許可を持っている--target-bucket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、「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

Valkey から既存のバックアップを新しい Valkey キャッシュまたは独自設計クラスターに復元し、既存の Redis OSSバックアップを新しい Redis OSSキャッシュまたは独自設計クラスターに復元できます。また、既存の Memcached サーバーレスキャッシュバックを新しい Memcached サーバーレスキャッシュに復元することもできます。

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、Valkey 7.2 以降と互換性のあるRDBファイル、および 5.0 から利用可能な最新バージョンまでの Redis OSSバージョン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、Valkey 7.2 以降と互換性のあるRDBファイル、および 5.0 から利用可能な最新バージョンまでの Redis OSSバージョン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
```

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)

Valkey または Redis OSS (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 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
  ```

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)

Valkey または Redis OSSサーバーレスキャッシュバックアップを復元したり、Valkey または Redis OSSが独自に設計したクラスターを復元したりできます。

Valkey または Redis OSS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、単一ノードの Valkey または Redis OSS (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) を使用して Valkey または Redis OSS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 Valkey または Redis OSS (クラスターモードが無効) または Valkey または Redis OSS (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、復元には Valkey または Redis OSS.rdb ファイルを使用します。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、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」を参照してください。

Valkey または Redis OSS (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、単一ノードの Valkey または Redis OSS (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) を使用して Valkey または Redis OSS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 Valkey または Redis OSS (クラス
  ```

ターモードが無効) または Valkey または Redis OSS (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、復元には Valkey または Redis OSS.rdb ファイルを使用します。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、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」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 は、バックアップが自動または手動で作成されたかどうかにかかわらず、いつでもバックアップを削除できる削除APIオペレーションを提供します。(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)。

ElastiCache コンソール、、 AWS CLIまたは を使用してバックアップを削除できます ElastiCache API。

バックアップの削除 (コンソール)

次の手順では、 ElastiCache コンソールを使用してバックアップを削除します。

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、次のパラメータで 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delete-serverless-cache-snapshot」の「」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)

以下のパラメータで delete-snapshot AWS CLI オペレーションを使用して、独自設計の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、請求書の費用をタグ値でグループ化することで、複数の AWS 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、 AWS CLI、または ElastiCache API を使用して、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、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。

チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする

新しい Valkey または Redis OSSの独自設計クラスターを作成するときは、Valkey または Redis OSS.rdb バックアップファイルからのデータでシードできます。クラスターのシードは、現在 の外部で Valkey または Redis OSSインスタンスを管理 ElastiCache していて、新しい ElastiCache (Redis OSS) 自己設計型クラスターに既存の Valkey または Redis データを入力する場合に便利です OSS。

Amazon 内で作成された Valkey または Redis OSSバックアップから新しい Valkey または Redis OSSのセルフ設計クラスターをシードするには 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。

Valkey または Redis OSS.rdb ファイルを使用して新しいセルフデザインクラスターをシードする場合、以下を実行できます。

- パーティション化されていないクラスターから、Redis OSSバージョン 3.2.4 を実行する Valkey または Redis OSS (クラスターモードが有効) のセルフ設計型クラスターにアップグレードします。
- 新しい独自設計クラスター内のシャード (APIおよび のノードグループと呼ばれますCLI) の数を指定します。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ー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ー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ーで使用されているノードタイプよりも大きいか小さいかを指定します。より小さなノードタイプにスケールする場合は、新しいノードタイプにデータおよび Valkey または Redis オーバーOSSヘッド用の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、「Valkey または Redis OSS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を確認する」を参照してください。
- 新しい Valkey または Redis OSS (クラスターモードが有効) クラスターのスロットのキーを、バックアップファイルの作成に使用されたクラスターとは異なる方法で分散します。

> **Note**
>
> Valkey または Redis OSS (クラスターモードが無効) クラスターから作成された .rdb ファイルから Valkey または Redis OSS (クラスターモードが有効) クラスターをシードすることはできません。

Important

- Valkey または Redis OSSバックアップデータがノードのリソースを超えないようにする必要があります。例えば、5 GB の Valkey または Redis OSS データを含む .rdb ファイルを、2.9 GB のメモリを持つ cache.m3.medium ノードに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 OSS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。
- Valkey または Redis .rdb OSS 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。

次に、Valkey または Redis ElastiCache の外部から ElastiCache (Redis) OSSにクラスターを移行する方法について説明しますOSS。

への移行 ElastiCache (Redis OSS)

- ステップ 1: Valkey または Redis OSS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

外部サービスから ElastiCache (Redis OSS) への移行。

- ステップ 1: Valkey または Redis OSS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: Valkey または Redis OSSバックアップを作成する

ElastiCache (Redis OSS) インスタンスにシードするための Valkey または Redis OSSバックアップを作成するには

1. 既存の Valkey または Redis OSSインスタンスに接続します。
2. BGSAVE 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、Valkey ウェブサイトBGSAVEの「」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、Valkey ウェブサイトSAVEの「」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、Valkey ウェブサイトの「永続性」を参照してください。

ステップ 2: Amazon S3 バケットとフォルダを作成する

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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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、新しい ElastiCache (Redis OSS) クラスターと同じ 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 ファイルを読み ElastiCache 取るときに、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、アクセス許可は、バケットとそのコンテンツを使用して新しい Valkey または Redis OSSクラスターをシードすることを許可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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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: Valkey または Redis OSS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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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 読み取りアクセスを許可します。バケットがデフォルトの 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、バックアップファイル ElastiCache へのアクセスを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 リージョンでの作業はすぐに開始できます。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、「デフォルトのリージョンで .rdb ファイルへの ElastiCache 読み取りアクセスを付与する」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。

デフォルトのリージョンで .rdb ファイルへの ElastiCache 読み取りアクセスを付与する

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 リージョンでの作業はすぐに開始できます。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。

デフォルトでは有効になっている 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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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. [Permissions] (許可) を選択します。
6. aws-scs-s3 読み取り専用またはIDs次のリストの正規のいずれかがユーザーとしてリストされていない場合は、以下を実行します。

   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、次のように 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 Valkey または Redis OSS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 デフォルトでは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)
- オブジェクトの読み取り/書き込みACLアクセス許可

d. [Save] 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 リージョンでの作業はすぐに開始できます。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 読み取りアクセスを許可します。

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、 で Amazon S3 コンソールを開きますhttps://console.aws.amazon.com/s3/。
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. [Save changes] (変更の保存) をクリックします。

.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、「Valkey または Redis 用のクラスターの作成 OSS」または「Valkey または Redis OSSレプリケーショングループをゼロから作成する」の手順に従います。クラスターエンジンOSSとして Valkey または Redis を必ず選択してください。

Amazon S3 にアップロードしたバックアップ ElastiCache の場所を知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。

（ ElastiCache Redis OSS) 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す。

- 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 S3 の場所ボックスに、ファイルの Amazon S3 パス (複数可) 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は、 パラメータを使用して`--snapshot-arns`、各 .rdb ファイルARNに対して完全修飾を指定します。例えば、`arn:aws:s3:::`*myBucket*/*myFolder*/*myBackupFilename*.rdb と指定します。は、Amazon S3 に保存したバックアップファイルに解決ARNする必要があります。

- の使用 ElastiCache API

  `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ARNに対して完全修飾を指定します。例えば、`arn:aws:s3:::`*myBucket*/*myFolder*/*myBackupFilename*.rdb と指定します。は、Amazon S3 に保存したバックアップファイルに解決ARNする必要があります。

> **Important**
>
> Valkey または Redis OSS (クラスターモードが有効) クラスターをシードするときは、新しい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create-replication-groupの：
> - APIリファレンスCreateReplicationGroupの ElastiCache API：

クラスターの作成プロセス中に、Valkey または Redis OSS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 を参照して、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 を使用して ElastiCache API、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。

でのエンジンバージョンとアップグレード ElastiCache

このセクションでは、サポートされている Valkey、Redis、OSSおよび Memcached エンジンとアップグレード方法について説明します。Redis OSS7.2 で使用できるすべての機能は、デフォルトで Valkey 7.2 以降で利用可能であることに注意してください。Redis OSS エンジン ElastiCache を搭載した既存の一部のエンジンから Valkey エンジンにアップグレードすることもできます。

トピック

のバージョン管理 ElastiCache

Valkey、Redis、および Memcached エンジン用に更新された ElastiCache キャッシュOSSと独自設計クラスターを更新する方法を管理します。

ElastiCache Serverless Cache のバージョン管理

ElastiCache Serverless キャッシュがアップグレードされるかどうかとタイミングを管理し、独自の条件とタイムラインでバージョンアップグレードを実行します。

ElastiCache Serverless 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、最新バージョンMINORとPATCH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MAJORバージョンが利用可能になると、 ElastiCache Serverless はコンソールに通知を送信し、 のイベントを送信します EventBridge。コンソール、、または を使用してキャッシュを変更APIし、最新のエンジンバージョンを選択することでCLI、キャッシュを最新バージョンにアップグレードできます。

独自設計 ElastiCache クラスターのバージョン管理

独自設計の ElastiCache クラスターを使用する場合、キャッシュクラスターを使用するソフトウェアが でサポートされている新しいバージョンにアップグレードされるタイミングを制御できます ElastiCache 。キャッシュを利用可能な最新の MAJOR、、MINORおよび PATCH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強化するプロトコル準拠のソフトウェアが、 でサポートされている新しいバージョンにアップグレードされるかどうか、およびいつアップグレードされるかを制御できます ElastiCache。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。

Valkey クラスターと Redis OSSクラスター

> **Note**
>
> - Valkey または Redis OSSクラスターが 1 つ以上のリージョンにレプリケートされている場合、エンジンバージョンはセカンダリリージョンでアップグレードされ、次にプライマリリージョンでアップグレードされます。
> - ElastiCache (Redis OSS) バージョンは、 MAJOR および MINOR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 OSS では、メジャーバージョンは 6、マイナーバージョン 2 です。独自設計のクラスターを操作すると、 ElastiCache (Redis OSS) は Redis OSS6.2.1 などのPATCHコンポーネントも公開し、パッチバージョンは 1 です。
>
>   MAJOR バージョンは互換性APIのない変更用であり、MINOR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。PATCH バージョン は、下位互換性のあるバグ修正と非機能的な変更用です。

Valkey と Redis ではOSS、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新しいエンジン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へのエンジンバージョン

アップグレードを開始します。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。

Memcached

Memcached では、新しいバージョンにアップグレードするには、キャッシュクラスターを変更し、使用する新しいエンジンバージョンを指定する必要があります。新しい Memcached バージョンへのアップグレードは破壊的な手順です – データは失われ、コールドキャッシュを使って開始されます。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。

古いバージョンの Memcached から Memcached バージョン 1.4.33 以降へアップグレードするときは、次の要件に注意する必要があります。以下の条件では、`CreateCacheCluster` および `ModifyCacheCluster` は失敗します。

- `slab_chunk_max > max_item_size` の場合。
- `max_item_size modulo slab_chunk_max != 0` の場合。
- `max_item_size > ((max_cache_memory - memcached_connections_overhead) / 4)` の場合。

  `(max_cache_memory - memcached_connections_overhead)` の値は、データに使用可能なノードのメモリです。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。

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す。これらは ElastiCache Serverless には適用されません。

Valkey と Redis に関するOSS考慮事項

独自設計の Valkey または Redis OSSクラスターをアップグレードする場合は、次の点を考慮してください。

- エンジンのバージョニングは、パッチの適用方法をできる限り制御できるように設計されています。ただし、 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し

た場合に備えて、ユーザーに代わってクラスターにパッチを適用する権利を ElastiCache 留保します。

- Valkey 7.2 および Redis 6.0 以降、 ElastiCache は複数のパッチバージョンを提供するのではなく、マイナーリリースごとに 1 OSS つのバージョンを提供します。
- Redis OSS 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伝播中にフェイルオーバー時間が長くなる場合があります (30 秒 ~ 1 分) 。
- Redis OSS7 以降、 は Valkey または Redis OSS (クラスターモードが無効) と Valkey または Redis OSS (クラスターモードが有効) の切り替え ElastiCache をサポートしています。
- Amazon ElastiCache (Redis OSS) エンジンのアップグレードプロセスは、既存のデータを保持するために最善を尽くすように設計されており、Redis OSSレプリケーションを成功させる必要があります。
- エンジンをアップグレードすると、 ElastiCache は既存のクライアント接続を終了します。エンジンアップグレード中のダウンタイムを最小限に抑えるには、エラー再試行と指数バックオフを含むRedis OSS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きに、Valkey または Redis OSS (クラスターモードが無効) から Valkey または Redis OSS (クラスターモードが有効) に直接アップグレードすることはできません。次の手順では、Valkey または Redis OSS (クラスターモードが無効) から Valkey または Redis OSS (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ます。

  Valkey または Redis OSS (クラスターモードが無効) から Valkey または Redis OSS (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. Valkey または Redis OSS (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 OSS (ノードグループ) を持つ Valkey または 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、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」を参照してください。

3. 古い Valkey または Redis OSS (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、でのクラスターの削除 ElastiCache 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 Valkey または Redis OSS (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) の数にスケールします。詳細については、「Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)」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。
- マルチ AZ が無効になっている単一の Redis OSSクラスターおよびクラスターについては、「」で説明されているOSSように、Redis が十分なメモリを使用できるようにすることをお勧めしますValkey または Redis OSS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を確認する。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 OSSクラスターの場合、受信書き込みトラフィックが少ない時間帯にエンジンのアップグレードをスケジュールすることをお勧めします。Redis OSS5.0.6 以降にアップグレードする場合、プライマリクラスターはアップグレードプロセス中にサービスリクエストで引き続き使用できます。

  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。

Memcached に関する考慮事項

独自設計の Memcached クラスターをアップグレードする場合は、次の点を考慮してください。

- エンジンのバージョニングは、パッチの適用方法をできる限り制御できるように設計されています。ただし、 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した場合に備えて、ユーザーに代わってクラスターにパッチを適用する権利を ElastiCache 留保します。
- Memcached エンジンでは永続性がサポートされていないため、そのエンジンバージョンのアップグレードは常に、クラスターのすべてのキャッシュデータを消去する破壊的なプロセスです。

エンジンバージョンのアップグレード方法

Valkey と Redis OSS

Valkey と Redis ではOSS、 ElastiCache コンソール、、または を使用してクラスターまたはレプリケーショングループを変更し、新しいエンジンバージョンを指定することで AWS CLI、クラスターまたは ElastiCache APIレプリケーショングループのバージョンアップグレードを開始します。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 ElastiCache AWS Management Console	の使用 AWS Management Console
AWS CLI での の使用 ElastiCache	の使用 AWS CLI
の使用 ElastiCache API	の使用 ElastiCache API

Memcached

Memcached では、クラスターのバージョンアップグレードを開始するには、クラスターを変更して新しいエンジンバージョンを指定します。これを行うには、 ElastiCache コンソール、 AWS CLI、または を使用します ElastiCache API。

- を使用するには AWS Management Console、 - を参照してくださいの使用 ElastiCache AWS Management Console。
- を使用するには AWS CLI、「」を参照してくださいAWS CLI での の使用 ElastiCache。
- を使用するには ElastiCache API、「」を参照してくださいの使用 ElastiCache API。

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

コンソールまたは を使用して、既存の Redis OSSレプリケーショングループ (v4 以降) を Valkey エンジンにアップグレードできますAPICLI。

> **Note**
>
> 既存の Redis OSS (クラスターモードが無効) シングルノードクラスターを Valkey エンジンにアップグレードする場合は、まず、前提条件のステップ に従ってレプリケーショングループに追加する必要がありますに既存の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。

デフォルトのキャッシュパラメータグループを使用している既存の Redis OSSレプリケーショングループがある場合は、 で modify-replication-group新しいエンジンとエンジンバージョンを指定することで Valkey にアップグレードできますAPI。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --engine valkey \
   --engine-version 7.2
```

Windows の場合:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --engine valkey ^
   --engine-version 7.2
```

アップグレードする既存の redis レプリケーショングループに適用されるカスタムキャッシュパラメータグループがある場合は、リクエストでカスタム Valkey キャッシュパラメータグループも渡す必要があります。入力 Valkey カスタムパラメータグループは、既存の Redis カスタムパラメータグループと同じ Redis 静的パラメータ値を持つ必要があります。

Linux、macOS、Unix の場合:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --engine valkey \
   --engine-version 7.2 \
   --cache-parameter-group-name myParamGroup
```

Windows の場合:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --engine valkey ^
   --engine-version 7.2 ^
   --cache-parameter-group-name myParamGroup
```

ElastiCache Serverless の Redis から Valkey OSSへのクロスエンジンアップグレード

で新しいエンジンとメジャーエンジンのバージョン modify-serverless-cacheを指定することでCLI、コンソールAPIまたは を使用して、既存の Redis OSSサーバーレスキャッシュを Valkey エンジンにアップグレードできますAPI。

Linux、macOS、Unix の場合:

```
aws elasticache modify-serverless-cache \
   --serverless-cache-name myCluster \
   --engine valkey \
   --major-engine-version 7
```

Windows の場合:

```
aws elasticache modify-serverless-cache ^
   --serverless-cache-name myCluster ^
   --engine valkey ^
   --major-engine-version 7
```

サポートされているエンジンとバージョン

ElastiCache サーバーレスキャッシュは、Valkey 7.2 以降、Redis OSSバージョン 7.0、および Memcached 1.6 以降をサポートしています。

ElastiCache 自己設計キャッシュは、Valkey 7.2 以降、すべての Redis OSSバージョン 4.0.10 以降、および Memcached バージョン 1.4.5 以降をサポートしています。

独自設計の ElastiCache クラスターは、次の Valkey バージョンをサポートしています。

- サポートされている Valkey バージョン
- サポートされている Redis OSSバージョン
- Redis OSSバージョンの有効期限終了スケジュール
- サポートされている ElastiCache (Memcached) バージョン

サポートされている Valkey バージョン

以下でサポートされている Valkey バージョン。Valkey は、デフォルトで Redis 7.2 OSS で使用できるほとんどの機能をサポートしています。

ElastiCache （ バルキー) バージョン 7.2.6

2024 年 10 月 10 日、Valkey 7.2.6 ElastiCache がリリースされました。Valkey 7.2 で導入された新機能の一部を以下に示します (Redis 7.1 OSS と比較して)。

- ZRANK および ZREVRANK コマンドの新しいWITHSCOREオプション
- CLIENT NO-TOUCH クライアントがキーの LRU/LFU に影響を与えずにコマンドを実行する場合。
- レプリケーションに基づいてクラスターモードでノードを論理的にグループ化するためにノードのシャード ID を返CLUSTERMYSHARDIDす新しいコマンド。
- さまざまなデータ型のパフォーマンスとメモリの最適化。

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

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

- キーが存在しなくなったときに解放されるブロックされたストリームコマンドには、別のエラーコード (- NOGROUPではなく -WRONGTYPE または -) が伴いますUNBLOCKED。
- コマンド統計は、コマンドが実際に実行された場合にのみ、ブロックされたコマンドに対して更新されます。
- ACL ユーザーの内部ストレージは、冗長なコマンドルールとカテゴリルールを削除しなくなります。これにより、これらのルールが ACL SAVE、、ACLGETUSERおよび ACL の一部として表示される方法が変わる可能性がありますLIST。
- TLSベースのレプリケーション用に作成されたクライアント接続はSNI、可能であれば使用します。
- XINFO STREAM: 表示時間レスポンスフィールドは、最後に成功したインタラクションではなく、最後に試行されたインタラクションを示すようになりました。新しいアクティブタイムレスポンスフィールドは、最後に成功したインタラクションを示すようになりました。
- XREADGROUP と X[AUTO〕CLAIM は、読み取り/クレームを実行できたかどうかにかかわらず、コンシューマーを作成します。〔TBD - ここで「it」とは〕
- ACL LIST/ で新しく作成されたデフォルトのユーザーセット sanitize-payload ACL フラグ GETUSER。
- HELLO コマンドは、成功しない限りクライアントの状態には影響しません。
- NAN レスポンスは、inf の現在の動作と同様に、単一の nan 型に正規化されます。

Valkey の詳細については、「Valkey」を参照してください。

Valkey 7.2 リリースの詳細については、 の「Redis OSS7.2.4 リリースノート (Valkey 7.2 には Redis からバージョン 7.2.4 OSSまでのすべての変更が含まれます) 」および「 Valkey 7.2 リリースノート」を参照してください GitHub。

サポートされている Redis OSSバージョン

ElastiCache サーバーレスキャッシュとセルフデザインキャッシュは、すべての Redis OSSバージョン 7.1 以前をサポートしています。

- ElastiCache （ バルキー) バージョン 7.2.6
- ElastiCache (Redis OSS) バージョン 7.1 (拡張)

独自設計の ElastiCache クラスターは、以下の Valkey および Redis OSSバージョンをサポートしています。

- ElastiCache (Redis OSS) バージョン 7.1 (拡張)
- ElastiCache (Redis OSS) バージョン 7.0 (拡張)
- ElastiCache (Redis OSS) バージョン 6.2 (拡張)
- ElastiCache (Redis OSS) バージョン 6.0 (拡張)
- ElastiCache (Redis OSS) バージョン 5.0.6 (拡張)
- ElastiCache (Redis OSS) バージョン 5.0.5 (廃止、バージョン 5.0.6 を使用)
- ElastiCache (Redis OSS) バージョン 5.0.4 (廃止、バージョン 5.0.6 を使用)
- ElastiCache (Redis OSS) バージョン 5.0.3 (廃止、バージョン 5.0.6 を使用)
- ElastiCache (Redis OSS) バージョン 5.0.0 (廃止、バージョン 5.0.6 を使用)
- ElastiCache (Redis OSS) バージョン 4.0.10 (拡張)
- 過去の製品終了 (EOL) バージョン (3.x)
- 過去の製品終了 (EOL) バージョン (2.x)

ElastiCache (Redis OSS) バージョン 7.1 (拡張)

このリリースには、ワークロードがより高いスループットと低いオペレーションレイテンシーを推進できるようにするパフォーマンスの向上が含まれています。 ElastiCache 7.1 では、次の 2 つの主な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口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レイヤーとは、クライアント入力を読み取るだけでなく、入力を Redis OSSバイナリコマンド形式に解析する拡張 I/O スレッドを意味します。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 OSS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-based `R7g.4xlarge` 以降 ElastiCache で実行する場合、お客様はノードあたり 1 秒あたり 100 万件を超えるリクエストを達成できます。 ElastiCache (Redis OSS) v7.1 のパフォーマンス向上により、お客様は ElastiCache (Redis OSS) v7.0 と比較してスループットを最大 100% 向上させ、P99 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、少なくとも 8 つの物理コアを持つノードサイズ (`2xlarge`Graviton および x86) `4xlarge`で有効になり、クライアントの変更は必要ありません。

> **Note**
>
> ElastiCache v7.1 は Redis v7.0 OSS と互換性があります。

ElastiCache (Redis OSS) バージョン 7.0 (拡張)

ElastiCache (Redis OSS) 7.0 では、新機能のいくつかの改善とサポートが追加されました。

- Functions : ElastiCache (Redis OSS) 7 は、Redis OSS Functions のサポートを追加し、クライアントが接続ごとにLUAスクリプトをサーバーに再送信することなく、開発者が ElastiCacheクラスターに保存されたアプリケーションロジックでスクリプトを実行できるようにするマネージドエクスペリエンスを提供します。
- ACL 改善点: Valkey と Redis 7 では、アクセスコントロールリストの次のバージョン () OSS のサポートが追加されましたACLs。クライアントは、Valkey および Redis の特定のキーまたはキースペースに複数のアクセス許可セットを指定できるようになりましたOSS。
- Sharded Pub/Sub : ElastiCache with Valkey と Redis 7 OSS では、クラスターモードで実行が有効になっている場合 ()、Pub/Sub ElastiCache 機能をシャーディングされた方法で実行するためのサポートが追加されていますCME。Pub/Sub 機能を使用すると、パブリッシャーはチャンネル上の任意の数のサブスクライバーにメッセージを発行できます。チャネルは ElastiCache クラスター内のシャードにバインドされるため、シャード間でチャネル情報を伝達する必要がなくなり、スケーラビリティが向上します。
- Valkey と Redis OSS7 ElastiCache による拡張 I/O マルチプレクシングにより、 ElastiCache クラスターへの多数のクライアント同時接続を持つ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、Redis OSS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オペレーション) が最大 72% 増加し、P99 レイテンシー ElastiCache が最大 71% 減少します。

Valkey の詳細については、「Valkey 」を参照してください。Redis OSS7.0 リリースの詳細については、 の Redis OSS 7.0 リリースノートを参照してくださいOSS GitHub。

ElastiCache (Redis OSS) バージョン 6.2 (拡張)

ElastiCache (Redis OSS) 6.2 には、8 vCPUs つ以上の x86 ノードタイプ、または 4 vCPUs つ以上の Graviton2 ノードタイプを使用する TLS対応クラスターのパフォーマンス向上が含まれています。これらの機能強化により、暗号化を他の にオフロードすることで、スループットが向上し、クライアント接続の確立時間が短縮されますvCPUs。Redis OSS6.2 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、ローカルにアタッチされた を含むクラスターノードのデータ階層化のサポートも導入しましたNVMeSSD。詳細については、「のデータ階層化 ElastiCache」を参照してください。

Redis OSS エンジンバージョン 6.2.6 では、Redis OSSクラスター内で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ある、ネイティブ JavaScript Object Notation (JSON) 形式もサポートされています。JSON サポートにより、 経由で動作するアプリケーションOSSAPIsのパフォーマンスと Redis を活用できますJSON。詳細については、「 の開始方法JSON」を参照してください。また、このデータタイプの使用状況をモニタリング CloudWatch するために に組み込まれJsonBasedCmdsLatencyている JSON関連のメトリクスJsonBasedCmdsや も含まれています。詳細については、「Valkey と Redis のメトリクス OSS」を参照してください。

6.2. ElastiCache （ Redis OSS) は、利用可能な Redis 6.2 OSS の優先パッチバージョン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/変更時に現在利用可能な Redis 6.2 OSS の優先パッチバージョンで起動されます。でエンジンバージョン 6.x を指定するAPIと、Redis OSS6 の最新バージョンになります。

既存の 6.0 クラスターでは、`CreateCacheCluster`、、`CreateReplicationGroup`または `ModifyReplicationGroup` `yes`で `AutoMinorVersionUpgrade`パラメータを に設定することで`ModifyCacheCluster`、次の自動マイナーバージョンアップグレードにオプトインできますAPIs。ElastiCache (Redis OSS) 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 でのセルフサービスの更新 ElastiCache」を参照してください。

を呼び出すと DescribeCacheEngineVersions API、`EngineVersion`パラメータ値は 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 OSS6.2 リリースの詳細については、 の Redis OSS 6.2 リリースノートを参照してください OSS GitHub。

ElastiCache (Redis OSS) バージョン 6.0 (拡張)

Amazon ElastiCache (Redis OSS) では、ロールベースのアクセス制御によるユーザーの認証、クライアント側のキャッシュ、大幅な運用上の改善など、Redis OSS エンジンの次のバージョンが導入されました。

Redis OSS6.0 以降、 ElastiCache (Redis OSS) は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 つのバージョンを提供します。 ElastiCache （ Redis OSS) 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、パフォーマンスとセキュリティを向上させます。

また、 AutoMinorVersionUpgradeパラメータを に設定することで、次の自動マイナーバージョンアップグレードにオプトインできます ElastiCache 。 yes (Redis OSS) はセルフサービスの更新を通じてマイナーバージョンアップグレードを管理します。詳細については、「のサービス更新 ElastiCache」を参照してください。

エンジンバージョンは、 ElastiCache (を使用して指定します6.0。Redis OSS) は、利用可能な Redis 6.0 OSS の優先パッチバージョン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/変更時に現在利用可能な Redis 6.0 OSS の優先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。

を呼び出すと DescribeCacheEngineVersions API、EngineVersionパラメータ値は 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 OSS6.0 リリースの詳細については、 の Redis OSS 6.0 リリースノートを参照してください OSS GitHub。

ElastiCache (Redis OSS) バージョン 5.0.6 (拡張)

Amazon ElastiCache (Redis OSS) では、次のバージョンの Redis OSS エンジンが導入されています。これには、バグ修正と以下の累積更新が含まれます。

- 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: FLUSHALLや などの偶発的なデータ損失を引き起こす可能性のある潜在的に危険または高価な Redis OSS コマンドの名前を変更rename-commandsできる というパラメータFLUSHDB。これは、オープンソース Redis の rename-command 設定に似ています OSS。ただし、 ElastiCache はフルマネージドワークフローを提供することでエクスペリエンスを向上させました。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 OSS パラメータの変更」を参照してください。

- Redis OSS Streams: これにより、プロデューサーが新しい項目をリアルタイムで追加できるログデータ構造がモデル化されます。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、「ストリーム」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、「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 OSS パラメータの変更」を参照してください。
- 新しい Redis OSSメトリクス、StreamBasedCmds。
- Redis OSSノードのスナップショット時間はわずかに短縮されます。

Important

ElastiCache (Redis OSS) は、Redis OSS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。

- RESTORE 特定のキーの有効期限が切れている場合の不一致応答。
- XCLAIM コマンドは、間違った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 OSS エンジンバージョン 5.0.0 の ElastiCache (Redis OSS) サポートに含まれており、今後のバージョン更新で消費されます。

詳細については、 OSSの Redis OSS の Redis 5.0.6 リリースノートを参照してください GitHub。

ElastiCache (Redis OSS) バージョン 5.0.5 (廃止、バージョン 5.0.6 を使用)

Amazon ElastiCache (Redis OSS) は、Redis OSS エンジンの次のバージョンを導入します。これには、予定されているすべてのオペレーション中に自動フェイルオーバークラスターの ElastiCache (Redis OSS) のオンライン設定の変更が含まれます。クラスターがオンラインのままで受信リクエストを処理し続ける間、クラスターをスケールし、Redis OSS エンジンバージョンをアップグレードし、パッチとメンテナンスの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、 の Redis OSS の「Redis 5.0.5 リリースノートOSS」を参照してください GitHub。

ElastiCache (Redis OSS) バージョン 5.0.4 (廃止、バージョン 5.0.6 を使用)

Amazon ElastiCache (Redis OSS) は、Amazon でサポートされている Redis OSS エンジンの次のバージョンを導入します ElastiCache。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、 OSSの Redis OSS の Redis 5.0.4 リリースノートを参照してください GitHub。

ElastiCache (Redis OSS) バージョン 5.0.3 (廃止、バージョン 5.0.6 を使用)

Amazon ElastiCache (Redis OSS) は、Amazon ElastiCache がサポートする次のバージョンの Redis OSS エンジンを導入します。これにはバグ修正が含まれています。

ElastiCache (Redis OSS) バージョン 5.0.0 (廃止、バージョン 5.0.6 を使用)

Amazon ElastiCache (Redis OSS) では、Amazon でサポートされている Redis OSS エンジンの次のメジャーバージョンが導入されました ElastiCache。 ElastiCache （ Redis OSS) 5.0.0 では、以下の改善がサポートされています。

- Redis OSS Streams: これにより、プロデューサーが新しい項目をリアルタイムで追加できるログデータ構造がモデル化されます。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、「ストリーム」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、「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 OSS パラメータの変更」を参照してください。
- 新しい Redis OSSメトリクス、StreamBasedCmds。
- Redis OSSノードのスナップショット時間はわずかに短縮されます。

ElastiCache (Redis OSS) バージョン 4.0.10 (拡張)

Amazon ElastiCache (Redis OSS) では、Amazon でサポートされている Redis OSS エンジンの次のメジャーバージョンが導入されました ElastiCache。 ElastiCache （ Redis OSS) 4.0.10 では、以下の改善点がサポートされています。

- オンラインクラスターのサイズ変更と暗号化の両方を単一の ElastiCache (Redis OSS) バージョンで行います。詳細については、次を参照してください。
  - Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)
  - Valkey または Redis のオンライン再シャーディング OSS (クラスターモードが有効)
  - Amazon のデータセキュリティ ElastiCache
- 多数の新しいパラメータ。詳細については、「Redis 4.0.10 OSS 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、「 コマンド」 (「」で検索) を参照してくださいMEMO。
- 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サポート。 ElastiCache （ Redis OSS) はUNLINK、メインスレッドとは異なるスレッドで実行する FLUSHALL FLUSHDBや などの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 OSSメトリクス、ActiveDefragHits。詳細については、「Redis のメトリクス OSS」を参照してください。

Redis OSSバージョン 3.2.10 を実行している Valkey または Redis OSS (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 (Redis OSS) クラスターのサイズ変更と暗号化のサポートの比較

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 *	不可	はい	可能
転送時の暗号化 **	可能	いいえ	可能
保管時の暗号化 **	可能	いいえ	可能
* シャードを追加、削除、および負荷分散します。			
** Fed RAMP、HIPAA、および PCI DSS 準拠アプリケーションに必要です。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。			

過去の製品終了 (EOL) バージョン (3.x)

ElastiCache (Redis OSS) バージョン 3.2.10 (拡張)

Amazon ElastiCache (Redis OSS) では、Amazon でサポートされている Redis OSS エンジンの次のメジャーバージョンが導入されています ElastiCache。ElastiCache (Redis OSS) 3.2.10 では、受信 I/O リクエストを引き続き処理している間に、クラスターにシャードを追加または削除するためのオンラインクラスターのサイズ変更が導入されています。 ElastiCache （ Redis OSS) 3.2.10 ユーザーは、データを暗号化する機能を除いて、以前の Redis OSSバージョンのすべての機能を持っています。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 Redis OSS) 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 *	不可	可能
転送時の暗号化 **	可能	不可
保管時の暗号化 **	可能	不可
* シャードを追加、削除、および負荷分散します。		
** Fed RAMP、HIPAA、および PCI DSS 準拠アプリケーションに必要です。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。

- Valkey または Redis のオンライン再シャーディング OSS (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 (Redis OSS) バージョン 3.2.6 (拡張)

Amazon ElastiCache (Redis OSS) では、Amazon でサポートされている Redis OSS エンジンの次のメジャーバージョンが導入されています ElastiCache。ElastiCache (Redis OSS) 3.2.6 ユーザーには、以前の Redis OSSバージョンのすべての機能に加えて、データを暗号化するオプションがあります。詳細については、次を参照してください。

- ElastiCache 転送中の暗号化 (TLS)
- の保管時の暗号化 ElastiCache
- Amazon のコンプライアンス検証 ElastiCache

ElastiCache (Redis OSS) バージョン 3.2.4 (拡張)

Amazon ElastiCache (Redis OSS) バージョン 3.2.4 では、Amazon でサポートされている Redis OSS エンジンの次のメジャーバージョンが導入されています ElastiCache。ElastiCache (Redis OSS) 3.2.4 ユーザーには、以前の Redis OSSバージョンのすべての機能に加えて、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があります。次の表に以下の内容がまとめてあります。

Redis OSS3.2.4 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	不可	可能
地理空間インデックス作成	あり	可能
ノードタイプの変更	可能	はい *
レプリカの拡張	可能	はい *
スケールアウト	不可	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 OSS では、6 つのGEOコマンドによる地理空間インデックス作成がサポートされています。詳細については、Valkey Commands ページの Redis OSS GEO* コマンドドキュメント コマンド: (に対してフィルタリング) を参照してくださいGEO。GEO

Redis OSS3 のその他の機能の詳細については、Redis 3.2 OSS リリースノート および Redis OSS3.0 リリースノート を参照してください。

現在 ElastiCache 管理されている Valkey または Redis OSS (クラスターモードが有効) では、以下の Redis 3.2 OSS 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 は、次の Redis 3.2 OSS 管理コマンドを無効にします。

- `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 OSS パラメータの詳細については、「」を参照してくださいRedis 3.2.4 OSS パラメータの変更。

過去の製品終了 (EOL) バージョン (2.x)

ElastiCache (Redis OSS) 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 Redis OSSの改善点には、バグ修正と不正なメモリアクセスアドレスのログ記録が含まれます。詳細については、「Redis 2.8 OSS リリースノート」を参照してください。

ElastiCache (Redis OSS) 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 OSSの改善には、バグ修正が含まれています。詳細については、「Redis 2.8 OSS 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 OSS パラメータの詳細については、 ElastiCache 「 ユーザーガイドRedis OSS 2.8.23 (拡張) でパラメータを追加」の「」を参照してください。

ElastiCache (Redis OSS) 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 OSSの改善点は次のとおりです。

- 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。
- 2 つの新しい CloudWatch メトリクスのサポート。
  - `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動作におけるいくつかの重要なバグ修正。詳細については、「Redis 2.8 OSS 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でレプリケーションパフォーマンスを強化し、クラスターの安定性を向上させるため、ElastiCache レプリカ以外の はサポートされなくなりました。
- 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 OSS設定変数 `appendonly` および `appendfsync`は、Redis OSS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、PUBSUBクライアントに発生する可能性が最も高いです。

ElastiCache (Redis OSS) バージョン 2.8.21

バージョン 2.8.19 以降に追加された Redis OSSの改善には、バグ修正が多数含まれています。詳細については、「Redis 2.8 OSS リリースノート」を参照してください。

ElastiCache (Redis OSS) 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 OSSの改善点は次のとおりです。

- のサポート HyperLogLog。詳細については、「Redis OSS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古いデータをレプリカノードに送信しないようにするには、バックグラウンドセーブ (bgsave) 子プロセスが中止されると、マスターはSYNC失敗します。
- HyperLogLogBasedCommands CloudWatch メトリクスのサポート。詳細については、「Valkey と Redis のメトリクス OSS」を参照してください。

ElastiCache (Redis OSS) 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 OSSの改善点は次のとおりです。

- 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 (Redis OSS) バージョン 2.6.13

Redis OSSバージョン 2.6.13 は、Amazon ElastiCache (Redis) でOSSサポートされている Redis の初期バージョンでしたOSS。マルチ AZ は Redis 2.6.13 OSS ではサポートされていません。

Redis OSSバージョンの有効期限終了スケジュール

このセクションでは、古いメジャーバージョンが発表される際の有効期限 (EOL) 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**
>
> ElastiCache (Redis OSS) 5.0.0 から 5.0.5 までの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 EOL

ソフトウェアバージョン	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 日付
バージョン 3	3.2.4、3.2.6、3.2.10	バージョン 6.2 以上 **Note** US-ISO-EAST-1、US-ISO-WEST-1、および US-ISOB-EAST-1 リージョンの場合は、5.0.6 以降をお勧めします。	2023 年 7 月 31 日
バージョン	2.8.24、2.8.23、2.8.22、2.8.21、2.8.19、、2.8.12、2.8.6、2.6.13	バージョン 6.2 以上	2023 年 1 月 13 日

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 日付
2	Note US-ISO-EAST-1、US-ISO-WEST-1、および US-ISOB-EAST-1 リージョンの場合は、5.0.6 以降をお勧めします。	

サポートされている ElastiCache (Memcached) バージョン

ElastiCache では、次の Memcached バージョンと新しいバージョンへのアップグレードがサポートされています。新しいバージョンにアップグレードする場合、満たされない場合にアップグレードが失敗する条件について、細心の注意を払ってください。

ElastiCache Memcached バージョン用

- Memcached バージョン 1.6.22
- Memcached バージョン 1.6.17
- Memcached バージョン 1.6.12
- Memcached バージョン 1.6.6
- Memcached バージョン 1.5.16
- Memcached バージョン 1.5.10
- Memcached バージョン 1.4.34
- Memcached バージョン 1.4.33
- Memcached バージョン 1.4.24
- Memcached バージョン 1.4.14

- Memcached バージョン 1.4.5

Memcached バージョン 1.6.22

ElastiCache (Memcached) は、Memcached バージョン 1.6.22 のサポートを追加します。新機能は含まれていませんが、バグ修正と Memcached 1.6.18 からの累積更新が含まれています。

詳細については、「」の「Memcached」のReleaseNotes「1622」を参照してください GitHub。

Memcached バージョン 1.6.17

ElastiCache (Memcached) は、Memcached バージョン 1.6.17 のサポートを追加します。新機能は含まれていませんが、バグ修正と Memcached 1.6.17 からの累積更新が含まれています。

詳細については、「」の「Memcached」のReleaseNotes「1617」を参照してください GitHub。

Memcached バージョン 1.6.12

ElastiCache (Memcached) は、Memcached バージョン 1.6.12 と転送中の暗号化のサポートを追加します。また、バグ修正と Memcached 1.6.6 からの累積更新が含まれています。

詳細については、「」の「Memcached」のReleaseNotes「1612」を参照してください GitHub。

Memcached バージョン 1.6.6

ElastiCache (Memcached) は、Memcached バージョン 1.6.6 のサポートを追加します。新機能は含まれていませんが、Memcached 1.5.16 からのバグ修正と累積更新が含まれています。 ElastiCache (Memcached) には Extstore のサポートは含まれていません。

詳細については、「」の「Memcached」のReleaseNotes「166」を参照してください GitHub。

Memcached バージョン 1.5.16

ElastiCache for Memcached で Memcached バージョン 1.5.16 のサポートが追加されました。新機能は含まれていませんが、バグ修正と Memcached 1.5.14 および Memcached 1.5.15 からの累積更新が含まれています。

詳細については、「」の「Memcached 1.5.16リリースノート」を参照してください GitHub。

Memcached バージョン 1.5.10

ElastiCache for Memcached バージョン 1.5.10 では、以下の Memcached 機能がサポートされています。

- 自動スラブ再分散。
- murmur3 アルゴリズムによる高速なハッシュテーブル参照。
- セグメント化されたLRUアルゴリズム。
- LRU クローラーからバックグラウンド再利用メモリ。
- --enable-seccomp: コンパイル時オプション。

no_modern および inline_ascii_resp パラメータも導入されています。詳細については、「Memcached 1.5.10 パラメータの変更」を参照してください。

Memcached バージョン 1.4.34 ElastiCache 以降に追加された Memcached の改善点は次のとおりです。

- ASCII マルチゲット、CVE-2017-9951、 のクロールの制限などの累積修正metadumper。
- 接続制限時に接続を閉じることにより接続管理を改善。
- 1 MB を超えるアイテムサイズのアイテムサイズ管理を強化。
- アイテムあたりのメモリ要件を数バイト下げることにより、パフォーマンスとメモリオーバーヘッドを改善。

詳細については、「」の「Memcached 1.5.10リリースノート」を参照してください GitHub。

Memcached バージョン 1.4.34

ElastiCache の Memcached バージョン 1.4.34 では、バージョン 1.4.33 に新機能が追加されていません。バージョン 1.4.34 は、通常のリリースよりも大きなバグ修正リリースです。

詳細については、「」の「Memcached 1.4.34リリースノート」を参照してください GitHub。

Memcached バージョン 1.4.33

バージョン 1.4.24 以降に追加された Memcached の機能拡張には、以下が含まれます。

- 特定のスラブクラス、スラブクラスのリスト、またはすべてのスラブクラスのメタデータをダンプできる機能 詳細については、「Memcached 1.4.31 リリースノート」を参照してください。
- デフォルト値が 1 MB を超える大きなアイテムに対応 詳細については、「Memcached 1.4.29 リリースノート」を参照してください。
- 閉じる前にクライアントがアイドル状態だった時間を指定できる機能

クラスターを再起動せずに、Memcached で使用可能なメモリの量を動的に増やすことができる機能 詳細については、「Memcached 1.4.27 リリースノート」を参照してください。

- `fetchers`、`mutations`、`evictions` のログ記録がサポートされるようになりました。詳細については、「Memcached 1.4.26 リリースノート」を参照してください。
- 解放されたメモリは、再度グローバルのプールに戻し、新しいスラブクラスに割り当て直すことができます。詳細については、「Memcached 1.4.25 リリースノート」を参照してください。
- 複数のバグ修正。
- いくつかの新しいコマンドとパラメータ。リストについては、「Memcached 1.4.33 で追加されたパラメータ」を参照してください。

Memcached バージョン 1.4.24

バージョン 1.4.14 以降に追加された Memcached の機能拡張には、以下が含まれます。

- バックグラウンドプロセスを使用した最近使用されていない (LRU) 管理。
- ハッシュアルゴリズムとして使用する jenkins または murmur3 のオプションを追加しました。
- いくつかの新しいコマンドとパラメータ。リストについては、「Memcached 1.4.24 で追加されたパラメータ」を参照してください。
- 複数のバグ修正。

Memcached バージョン 1.4.14

バージョン 1.4.5 以降に追加された Memcached の機能拡張には、以下が含まれます。

- スラブ再分散機能の強化。
- パフォーマンスとスケーラビリティの強化。
- touch コマンドの導入により、既存の項目の有効期限を取得せずに更新する機能。
- 自動検出—クライアントプログラムが、クラスター内のすべてのキャッシュノードを自動的に識別し、それらのすべてのノードへの接続を開始して維持する機能。

Memcached バージョン 1.4.5

Memcached バージョン 1.4.5 は、Amazon ElastiCache (Memcached) でサポートされている最初のエンジンおよびバージョンでした。

Valkey とのメジャーバージョンの動作と互換性の違い

Valkey 7.2.6 は、以前のバージョンの Redis 7.2.5 OSS と互換性が似ています。Valkey の最新バージョンについては、「」を参照してくださいサポートされているエンジンとバージョン。

Redis とのメジャーバージョンの動作と互換性の違い OSS

> **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。
> シーケンシャルアップグレードを必要とせずに、現在の Redis OSSバージョンから利用可能な最新の Redis OSSバージョンに直接アップグレードできます。例えば、Redis OSS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 OSSバージョンは、、MAJOR、MINORおよび PATCH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 OSS では、メジャーバージョンは 4、マイナーバージョン 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。

- MAJOR バージョンは互換性APIのない変更用です
- MINOR バージョン は、下位互換の方法で追加された新機能用です。
- PATCH バージョン は、下位互換性のあるバグ修正と非機能的な変更用です。

最新のパフォーマンスと安定性を向上させるために、常に特定の . MINORバージョン内の最新のパッチMAJORバージョンを使用することをお勧めします。Redis OSS6.0 以降、 ElastiCache (Redis OSS) は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 つのバージョンを提供します。 ElastiCache （ Redis OSS) 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、パフォーマンスとセキュリティを向上させます。

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。が新しい AWS リージョンに可用性 ElastiCache を拡張すると、 ElastiCache (Redis OSS) は、その時点で新しいリージョンの 2 つの最新の MAJOR.MINOR バージョンをサポートします。例えば、新しい AWS リージョンが起動し、

最新の MAJOR.MINOR ElastiCache （ Redis OSS) バージョンは 7.0 および 6.2 で、 ElastiCache (Redis OSS) は新しい AWS リージョンでバージョン 7.0 および 6.2 をサポートします。新しい MAJOR。MINOR ElastiCache (Redis OSS) のバージョンがリリースされると、 ElastiCache は引き続き、新しくリリースされた ElastiCache (Redis OSS) バージョンのサポートを追加します。のリージョンの選択の詳細については ElastiCache、「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。

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実行するときは、動作や、時間のOSS経過とともに Redis でリリースされる下位互換性のない変更を含む次のリストを考慮してください。

Redis 7.0 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 7.0 OSS 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。スクリプトにある程度の耐久性が必要な場合は、Redis OSS関数 の使用を検討することをお勧めします。
- Pubsub チャネルは、新しいACLユーザーに対して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、、LASTSAVEREADONLYREADWRITE、 のACLカテゴリASKING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 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 6.2 OSS 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 OSS からアップグレードすると、マップレスポンスから lua スクリプトに返されるキーと値のペアの順序が変更されます。スクリプトがマップを使用`redis.setresp()`または返す場合 (Redis 6.0 OSS では新しい) 、アップグレード時にスクリプトが中断する可能性がある影響を考慮してください。

Redis 6.0 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 6.0 OSS 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` パラメータをはいに設定すると、 ElastiCache (Redis OSS) 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。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、「」の「セルフサービスの更新 ElastiCache」を参照してください。

Redis 5.0 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 5.0 OSS 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 (Redis OSS) 5.0 でのみ利用可能な前の動作に戻すオプションがあります。
- Redis 4.0 OSS 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 OSS4.0 では、Redis は応答を確定するために

一部の応答をレキソグラフィーで順序付けOSSします。この順序は、スクリプトがエフェクトによってレプリケートされる場合には適用されません。

- Inn Redis OSS 5.0.3 以降 ElastiCache (Redis OSS) では、一部の IO 作業が 4 を超える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されますVCPUs。これにより、Redis のパフォーマンス特性OSS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 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 4.0 OSS 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 EOL

Redis 3.2 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 3.2 OSS 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 OSSバージョンの有効期限終了スケジュール」を参照してください。

Redis 2.8 OSS の動作と下位互換性のない変更

変更の完全なリストについては、「Redis 2.8 OSS リリースノート」を参照してください。

- Redis OSS2.8.22 以降、Redis OSSAOFは ElastiCache (Redis) ではサポートされなくなりましたOSS。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 OSS2.8.22 以降、 ElastiCache (Redis OSS) は 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ていません ElastiCache。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。外部レプリカの代替として、Redis 6.0 OSS で使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 OSS および以前のバージョンは、両方の条件に対して -1 を返すために使用されます。
- 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 OSSバージョンの有効期限終了スケジュール」を参照してください。

ブロックされた Valkey または Redis OSS エンジンのアップグレードの解決

次の表に示すように、スケールアップオペレーションが保留中の場合、Valkey または Redis OSS エンジンのアップグレードオペレーションはブロックされます。

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

ブロックされた Redis OSS 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。
  - 即時適用チェックボックスをオフにして、次のメンテナンスウィンドウの Redis OSS エンジンアップグレードオペレーションをスケジュールします。

    ではCLI、 を使用します`--no-apply-immediately`。ではAPI、 を使用しますApplyImmediately=false。
  - 次のメンテナンスウィンドウ (または後) が Redis OSS エンジンのアップグレードオペレーションを実行するまで待ちます。

- このクラスターの変更に Redis OSSスケールアップオペレーションを追加し、すぐに適用チェックボックスをオンにします。

  ではCLI、 を使用します`--apply-immediately`。ではAPI、 を使用しますは`ApplyImmediately=true`。

  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。

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

- 全体的なベストプラクティス
- サポートされている Valkey、Redis、OSSおよび Memcached コマンド
- Valkey と Redis OSSの設定と制限
- IPv6 Valkey、Redis、OSSMemcached のクライアント例
- クライアントのベストプラクティス (Valkey と Redis OSS)
- クライアントのベストプラクティス (Memcached)
- TLS 有効なデュアルスタック ElastiCache クラスター
- Valkey と Redis の予約済みメモリの管理 OSS
- Valkey および Redis OSSの独自設計クラスターを使用する際のベストプラクティス
- Memcached のキャッシュ戦略

全体的なベストプラクティス

以下は、 内で Valkey、Redis、OSSおよび Memcached インターフェイスを使用するための最適な方法に関する情報です ElastiCache。

- クラスターモードを有効にした設定を使用する – クラスターモードを有効にした場合、キャッシュを水平方向にスケールして、クラスターモードが無効になった設定よりも高いストレージとスループットを実現できます。 ElastiCache サーバーレスはクラスターモードを有効にした設定でのみ使用できます。

- 長期接続の使用 – 新しい接続の作成にはコストがかかり、キャッシュから時間とCPUリソースがかかります。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。
- レプリカからの読み取り – ElastiCache サーバーレスまたはプロビジョニングされたリードレプリカ (セルフ設計クラスター) を使用している場合は、レプリカに読み取りを指示して、スケーラビリティの向上やレイテンシーの低減を実現します。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。

  独自設計型のクラスターでは、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) から読み取ると、可能な場合はクライアントのローカルアベイラビリティーゾーンに読み取りが転送され、取得レイテンシーが短縮されます。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。
- シャードパブ/サブを使用する – Valkey または Redis を使用してスループットの高いパブ/サブワークロードOSSをサポートする場合は、シャードパブ/サブ (Valkey および Redis 7 以降で利用可能) OSS を使用することをお勧めします。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。 ElastiCache サーバーレスでは、従来の pub/sub コマンドは内部的にシャード pub/sub コマンドを使用することに注意してください。

サポートされている Valkey、Redis、OSSおよび Memcached コマンド

サポートされている Valkey コマンドと Redis OSS コマンド

サポートされている Valkey コマンドと Redis OSS コマンド

以下の Valkey コマンドと Redis OSS 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 Valkey コマンドと Redis OSS 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

  Valkey または Redis OSSクラスターレプリカノードへの接続に対して読み取り専用クエリを有効にします。

  詳細はこちら

- READWRITE

  Valkey または Redis OSSクラスターレプリカノードへの接続の読み取り/書き込み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

  Valkey または Redis OSSサーバーでのハンドシェイク。

  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_RO

  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 (複数可) によって観測されたセット (複数可) の近似カーディナリティを返します。

  詳細はこちら

- PFMERGE

  1 つ以上の 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 コマンドは内部的にシャード を使用するためPUBSUB、チャネル名は混合されます。

- PUBLISH

  チャネルにメッセージを投稿します。

  詳細はこちら
- PUBSUB CHANNELS

  アクティブなチャネルを返します。

  詳細はこちら
- PUBSUB NUMSUB

  チャ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ネルを返します。

PUBSUB-SHARDCHANNELS

- PUBSUB SHARDNUMSUB

  シャードチャ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_RO

  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

  ダイSHA1ジェストによってサーバー側の Lua スクリプトを実行します。

  詳細はこちら

- EVALSHA_RO

  ダイSHA1ジェストによって読み取り専用のサーバー側の Lua スクリプトを実行します。

  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 ユーザーを識別するために使用できる擬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

  コンピュータアートと Valkey または Redis OSS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

  の範囲内のストリームからのメッセージを返しますIDs。

  詳細はこちら

- XREAD

  リクエストされたストリームよりもIDs大きい複数のストリームから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

  の範囲内のストリームからのメッセージを逆の順序IDs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

制限付き Valkey コマンドと Redis OSS コマンド

マネージドサービスエクスペリエンスを提供するために、 は、高度な権限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 ElastiCache を制限します。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 getredir
- 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

サポートされている Memcached のコマンド

ElastiCache Serverless for Memcached は、以下を除くオープンソース memcached 1.6 のすべての memcached コマンドをサポートします。

- クライアント接続には が必要です。その結果TLS、UDPプロトコルはサポートされていません。
- バイナリプロトコルは memcached 1.6 で正式に廃止されたため、サポートされていません。
- 大量のキーを取得することによるサーバーへの DoS 攻撃の可能性を回避するため、GET/GETS コマンドは 16KB に制限されています。
- 遅延した flush_all コマンドは CLIENT_ERROR で拒否されます。
- エンジンを設定したり、エンジンの状態やログに関する内部情報を公開したりする次のようなコマンドはサポートされていません。
  - STATS コマンドでは、stats と stats reset がサポートされます。他のバリエーションは ERROR を返します。
  - lru / lru_crawler - LRU LRUおよびクローラ設定の変更
  - watch - memcached のサーバーログをモニタリングします
  - verbosity - サーバーログレベルを設定します
  - me - メタデバッグ (me) コマンドはサポートされていません

Valkey と Redis OSSの設定と制限

Valkey エンジンと Redis OSS エンジンには、それぞれ多数の設定パラメータが用意されています。その中には ElastiCache (Redis OSS) 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の場合、パラメータグループは使用されず、すべての Valkey または Redis OSS設定は変更できません。次の Valkey または Redis OSSパラメータが設定されています。

名前	詳細	説明
acl-pubsub-default	allchannels	キャッシュ上のACLユーザーのデフォルトの pubsub チャネルアクセス許可。
client-output-buffer-limit	normal 0 0 0 pubsub 32mb 8mb 60	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 32MiB バックログに違反した場合、または 60 秒間 8MiB バックログに違反した場合、/ クライアントは切断されます。
client-query-buffer-limit	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リクエストを発行することはできません。
cluster-allow-pubsubshard-when-down	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 Pubsub トラフィックを処理できます。
cluster-allow-reads-when-down	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。
cluster-enabled	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。

名前	詳細	説明
cluster-require-full-coverage	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 cluster slots 内の 1 つの仮想ノードによって「カバー」されます。
lua-time-limit	5000	Lua スクリプトの最大実行時間をミリ秒単位で、 ElastiCache はスクリプトを停止するアクションを実行します。 lua-time-limit を超過すると、すべての Valkey または Redis OSS コマンドが ____-BUSY という形式のエラーを返す可能性があります。この状態は、多くの重要な Valkey または Redis OSSオペレーションに干渉する可能性があるため、 ElastiCache は最初にSCRIPTKILLコマンドを発行します。これが失敗した場合、 ElastiCache は Valkey または Redis を強制的に再起動しますOSS。
maxclients	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、成功することもあれば、失敗することもあります。
maxmemory-policy	volatile-lru	TTL セットを持つ項目は、キャッシュのメモリ制限に達したときに (LRU) 推定後に least-recently-used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。

名前	詳細	説明
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 Valkey または Redis OSSキーまたはチャネル名の最大サイズ。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 Valkey または Redis OSS Lua スクリプトの最大サイズ。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。
スロットサイズ	32 GiB	単一の Valkey または Redis OSSハッシュスロットの最大サイズ。1 つの Valkey または Redis OSS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クライアントは、スロットのエビクションポリシーをトリガーし、エビ

名前	詳細	説明
		ケート可能なキーがない場合、 はメモリ不足 (OOM) エラーを受け取ります。

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、「Valkey パラメータと Redis OSS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。

IPv6 Valkey、Redis、OSSMemcached のクライアント例

ElastiCache は Valkey、Redis、OSSおよび Memcached と互換性があります。つまり、IPv6接続をサポートするクライアントは、IPv6有効な ElastiCache (Memcached) クラスターに接続できる必要があります。IPv6 有効な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。

ElastiCache リソースの Valkey および Redis クライアントの設定に関する推奨事項については、データベースブログの「Valkey および Redis クライアントのベストプラクティス」ブログ記事を参照してください。 AWS OSS

以下は、一般的に使用されるオープンソースクライアントライブラリでIPv6有効な ElastiCache リソースを操作するためのベストプラクティスです。

Valkey と Redis で検証されたクライアント OSS

ElastiCache は Valkey およびオープンソース Redis と互換性がありますOSS。つまり、IPv6接続をサポートする Valkey およびオープンソースの Redis OSSクライアントは、IPv6有効な ElastiCache (Redis OSS) クラスターに接続できる必要があります。さらに、最も人気のある 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。

以下のクライアントは、Valkey および Redis でサポートされているすべてのネットワークタイプ設定で動作することが特に検証されていますOSS。

検証済みクライアント:

- Redis Py () – 4.1.2
- レタス – バージョン: 6.1.6。RELEASE

- Jedis – バージョン: 3.6.0

クライアントのベストプラクティス (Valkey と Redis OSS)

一般的なシナリオのベストプラクティスと、最も一般的なオープンソースの Valkey および Redis OSSクライアントライブラリ (redis-py、PHPRedis、Lettuce) のコード例、および一般的に使用されるオープンソースの Memcached クライアントライブラリと ElastiCache リソースを操作するためのベストプラクティスについて説明します。

トピック

- 多数の接続 (Valkey と Redis OSS)
- クラスタークライアント検出とエクスポネンシャルバックオフ (Valkey と Redis OSS)
- クライアント側のタイムアウトを設定する (Valkey と Redis OSS)
- サーバー側のアイドルタイムアウトを設定する (Valkey と Redis OSS)
- Lua スクリプト
- 大きな複合項目の保存 (バルキーと Redis OSS)
- Lettuce クライアント設定 (Valkey と Redis OSS)
- デュアルスタッククラスターの優先プロトコルの設定 (Valkey および Redis OSS)

多数の接続 (Valkey と Redis OSS)

サーバーレスキャッシュと個々の ElastiCache (Redis OSS) 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。Valkey と Redis OSSはそれぞれ、受信クライアントリクエストが順番に処理されるイベントループに基づいて、シングルスレッドプロセスがあります。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。

Valkey または Redis OSSサーバーで接続のボトルネックが発生しないように、次のアクションを実行できます。

- リードレプリカからの読み取り操作を実行する。これは、クラスターモードで ElastiCache リーダーエンドポイントを無効にするか、サーバーレスキャッシュを含むクラスターモードでの読み取りに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 Valkey または Redis OSSクラスターは、クラスターモード対応クライアントで使

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。

一般的に、TCP接続の作成は、一般的な Valkey または Redis OSS コマンドと比較して計算コストの高い操作です。例えば、SET/GET リクエストの処理は、既存の接続を再利用する場合に、桁違いに高速になります。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。

次のコード例は、新しいユーザーリクエストごとに新しい接続が作成されているPHPRedis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;
```

このコードを Graviton2 (m6g.2xlargeEC2 ） ElastiCache (Redis) ノードに接続された Amazon Elastic Compute Cloud (Amazon OSS) インスタンスのループでベンチマークしました。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);
}
```

クラスタークライアント検出とエクスポネンシャルバックオフ (Valkey と Redis OSS)

クラスターモードで ElastiCache Valkey または Redis OSSクラスターに接続する場合、対応するクライアントライブラリはクラスターを認識している必要があります。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。
- レプリカによって以前のプライマリノードによって処理されたすべてのスロットが引き継がれるフェイルオーバーの状況や、スロットがソースプライマリからターゲットプライマリノードに移動されるときに再シャーディングなど、サーバーからMOVEDリダイレクトを受信します。

クライアント検出は通常、Valkey または Redis OSSサーバーに CLUSTERSLOTまたは CLUSTER NODE コマンドを発行することで行われます。スロット範囲のセットと、関連付けられたプライマ

リノードとレプリカノードをクライアントに返すため、 CLUSTERSLOTメソッドをお勧めします。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。

クラスタートポロジによっては、CLUSTERSLOTコマンドのレスポンスのサイズがクラスターのサイズによって異なる場合があります。ノード数が多い大きなクラスターは、応答も大きくなります。したがって、クラスタートポロジー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ー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。これにより、Valkey または Redis OSSサーバーが長期間応答しなくなり、CPU使用率が 100% になる可能性があります。各CLUSTERSLOTコマンドがクラスターバス内の多数のノードを処理する必要がある場合、停止は延長されます。この動作により、Python (redis-py-cluster) や Java (Lettuce や Redisson) など、さまざまな言語にわたって、過去に複数のクライアント停止が発生しています。

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ートポロジが静的であり、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、クラスター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。
- 「での接続エンドポイントの検索 ElastiCache」のガイドを参考にして、クラスターエンドポイントを検索し、クラスター検出を実行する。これにより、クラスター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。

以下は、redis-py、、PHPRedisおよび Lettuce の指数バックオフ再試行ロジックのコード例です。

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 OSS オブジェクトの作成時に指定されたretry_on_timeout引数を使用して有効にできます。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サンプル 2: PHPRedis

PHPRedis には、最大 10 回 (設定不可) 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。エクスポネンシャルバックオフを PHPredis (#1986) に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、 がマージされて文書化されています。 の最新リリースの ではPHPRedis、手動で を実装する必要はありませんが、以前のバージョンの のリファレンスをここに含めました。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 (Valkey と Redis OSS)

クライアント側のタイムアウトの設定

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。特定の Valkey または Redis OSS コマンドは、他のコマンドよりも計算コストがかかる場合があります。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 スクリプトまたは 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 Valkey コマンドまたは Redis OSS 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
- などのブロッキングオペレーションの実行 BLPOP

などのブロッキングオペレーションの場合BLPOP、ベストプラクティスはコマンドタイムアウトをソケットタイムアウトよりも小さい数値に設定することです。

以下は、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ためのコード例です。

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)
```

タイムアウト設定サンプル 2: PHPRedis

以下は、 を使用したコード例ですPHPRedis。

```
// 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"); // BLPOP 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を設定する (Valkey と Redis OSS)

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、Valkey パラメータと Redis OSS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。

Lua スクリプト

Valkey と Redis は、Lua スクリプトを実行するコマンドを含め、200 を超えるコマンドOSSをサポートしています。ただし、Lua スクリプトについては、Valkey または Redis のメモリと可用性に影響を与える可能性のあるいくつかの落とし穴がありますOSS。

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 Valkey または Redis OSS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 スクリプトは一意であるため、Valkey または Redis OSSサーバーが多数の

Lua スクリプトを保存し、より多くのメモリを消費する可能性があります。これを軽減するには、すべての Lua スクリプトがパラメータ化され、必要に応じて定期的に実行SCRIPTFLUSHしてキャッシュされた Lua スクリプトを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 Valkey コマンドや Redis OSS コマンドよりも完了までに時間がかかる場合があります。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。長時間実行されている Lua スクリプトをミューテーションすると、Valkey または Redis OSS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

Lua スクリプトは、Valkey または Redis OSSが を超えているOSS場合でも、Valkey または Redis に新しいデータを書き込む方法をいくつか示しますmaxmemory。

- スクリプトは、Valkey または Redis OSSサーバーが を下回ったときに開始されmaxmemory、 内に複数の書き込みオペレーションが含まれます。
- スクリプトの最初の書き込みコマンドはメモリを消費せず (などDEL)、その後にメモリを消費する書き込みオペレーションが増えます。

- この問題は、 以外の Valkey または Redis OSSサーバーで適切な立ち退きポリシーを設定することで軽減できますnoeviction。これにより、Redis OSSはアイテムを削除し、Lua スクリプト間でメモリを解放できます。

大きな複合項目の保存 (バルキーと Redis OSS)

一部のシナリオでは、アプリケーションは大きな複合項目を Valkey または Redis OSS (マルチ GB ハッシュデータセットなど) に保存することがあります。これは、Valkey または Redis のパフォーマンス問題につながることが多いため、推奨されませんOSS。例えば、クライアントはHGETALLコマンドを実行してマルチ GB ハッシュコレクション全体を取得できます。これにより、クライアント出力バッファ内の大きな項目をバッファリングする Valkey または Redis OSSサーバーに大きなメモリ圧力が発生す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の場合、シリアル化されたサイズが 256 ElastiCache MB を超える項目を含むスロットは移行しません。

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、同じコマンド内の複数のキー値を取得できます。
- 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。1 つの例は、HGETALLハッシュコレクションではなく HGETまたは HMGET コマンドを使用することです。

Lettuce クライアント設定 (Valkey と Redis OSS)

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、それらが ElastiCache クラスターにどのように適用されるか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: クラスターモードの Lettuce 設定、TLS有効
- 例: クラスターモードのレタス設定が無効、TLS有効

Java DNSキャッシュ TTL

Java 仮想マシン (JVM) はDNS名前ルックアップをキャッシュします。がホスト名を IP アドレスにJVM解決すると、 time-to-live () と呼ばれる指定された期間、IP アドレスがキャッシュされますTTL。

TTL 値の選択は、レイテンシーと変化への応答性のトレードオフです。が短いとTTLs、DNSリゾルバーはクラスターの更新DNSに気づきます。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、クエリボリュームが増加し、アプリケーションのレイテンシーが長くなる可能性があります。正しいTTL値はありませんが、TTL値を設定するときに変更が有効になるまでに余裕がある時間を考慮する必要があります。

ElastiCache ノードは変更される可能性のあるDNS名前エントリを使用するため、 を 5 ~ 10 秒TTLの低さJVMに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ると、アプリケーションはDNSエントリを再クエリすることで、リソースの新しい IP アドレスを受信して使用できます。

一部の Java 設定では、 JVMが再起動されるまでDNSエントリが更新されないようにJVMデフォルトTTLが設定されます。

の設定方法の詳細についてはTTL、JVM「 の設定方法TTLJVM」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DNSルックアップは、クラスター内で使用可能なすべてのノードのリストURIを返し、クラスターの初期化中にランダムにそのうちの 1 つに解決されます。トポロジの更新の仕組みの詳細については、このトピックのdynamicRefreshResources後半を参照してください。

SocketOption

を有効にしますKeepAlive。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: クラスターモードを有効にするクライアントオプション

接続が失われAutoReconnectたときに を有効にします。

を設定しますCommandTimeout。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。

トポロジから障害が発生したノードを除外nodeFilterするように を設定します。Lettuce は、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) の「クラスターノード」出力 (PFAIL/FAIL ステータスのノードを含む) にあるすべてのノードを保存します。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終了し、クラスターが復旧プロセスを開始した後、 clusterTopology が更新されている間、クラスターバスノードマップには、ダウンノードがノードとしてリストされてからトポロジからFAIL完全に削除されます。この期間中、Lettuce クライアントは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、true DynamicRefreshSources 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。複数のシードノード (DynamicRefreshSources が true の場合) があると、少なくとも一部のシードノードで、新

しく昇格したプライマリでのフェイルオーバー後にトポロジビューが更新される必要があるため、この問題が発生する可能性が低くなります。

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の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ー検出機能と互換性があるわけではありません。で無効になっているクラスターモード ElastiCache は、Lettuce の と互換性がありませんMasterSlaveTopologyRefresh。代わりに、クラスタ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、「Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)」を参照してください。
> Lettuce の動的トポロジー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。

を有効にしますenablePeriodicRefresh。これにより、クラスタートポロジーの定期的な更新が可能になり、クライアントはクラスタートポロジーを の間隔 refreshPeriod (デフォルト: 60 秒) で更新します。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。

を有効にしますenableAllAdaptiveRefreshTriggers。これにより、_、MOVED_REDIRECT、ASK_REDIRECT、PERSISTENT_RECONNECTS、UNCOVERED_

のすべてのトリガーを使用するアダプティブトポロジの更新が可能になります SLOTUNKNOWNNODE。アダプティブ更新トリガーは、Valkey または Redis OSSクラスターオペレーション中に発生する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。

を有効にしますcloseStaleConnections。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。

を有効にしますdynamicRefreshResources。小さいクラスターでは を有効に dynamicRefreshResourcesし、大きいクラスターでは無効にすることをお勧めします。dynamicRefreshResources は、提供されたシードノード (クラスター設定エンドポイントなど) からクラスターノードを検出します。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と失敗し、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 で を設定しますDirContextDnsResolver。DNS リゾルバーは Java の com.sun.jndi.dns に基づいていますDnsContextFactory。

指数バックオフとフルジッターreconnectDelayを使用して を設定します。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の詳細については、 AWS データベースブログの「ベストプラクティス」ブログ記事の「バックオフロジック」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、、、、、またはLua スクリプトなど、複数のキーで反復するコマンドにはFLUSHALLKEYSSMEMBERS、より長いタイムアウトを使用します。、、 などの単一キーコマンドにはSETGET、短いタイムアウトを使用しますHSET。

> **Note**
>
> 次の例で設定されているタイムアウトは、キーと値が 20 バイトまでの SET/GET コマンドを実行するテスト用です。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: クラスターモードの Lettuce 設定、TLS有効

> **Note**
>
> 次の例のタイムアウトは、最大 20 バイト長のキーと値を使用して SET/GET コマンドを実行するテスト用です。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 optionts
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: クラスターモードのレタス設定が無効、TLS有効

> **Note**
>
> 次の例のタイムアウトは、最大 20 バイト長のキーと値を使用して SET/GET コマンドを実行するテスト用です。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);
```

デュアルスタッククラスターの優先プロトコルの設定 (Valkey および Redis OSS)

クラスターモードが有効になっている Valkey または Redis OSSクラスターの場合、IP Discovery パラメータを使用してクラスター内のノードへの接続に使用するプロトコルクライアントを制御できます。IP Discovery パラメータは、 IPv4または のいずれかに設定できますIPv6。

Valkey または Redis OSSクラスターの場合、IP 検出パラメータは、クラススロット ()、クラスターシャード ()、クラスターノード () 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IPsの 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。Valkey または Redis OSSクラスターに変更が完全に伝達されたタイミングを確認するには、 の出力をモニタリングしますcluster slots。クラスタースロット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IPsとともにレポートされると、変更の伝播は完了します。

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;
}));
```

クライアントのベストプラクティス (Memcached)

効率的なロードバランシングのための ElastiCache クライアントの設定 (Memcached)

> **Note**
>
> このセクションは、独自設計型のマルチノード Memcached クラスターに適用されます。

複数の ElastiCache Memcached ノードを効果的に使用するには、キャッシュキーをノード全体に分散できる必要があります。ノード数 n 個のクラスターで負荷を分散する場合、簡単な方法は、オブジェクトのキーのハッシュを計算し、その結果を n - hash(key) mod n で除算して剰余を求めることです。結果の値 (0 ~ n–1) が、オブジェクトを配置するノードの数になります。

この手法は単純で、ノードの数 (n) が一定である限り有効です。ただし、クラスターからノードを追加または削除する場合、移動する必要があるキーの数は (n - 1) / n (n は新しいノード数) です。したがって、この手法では多数のキーが移動され、特にノード数が大きくなると、初期のキャッシュミスが多数発生します。1 ノードから 2 ノードへのスケーリングでは、キーの (2-1)/2 (50 パーセント) が移動されます。9 ノードから 10 ノードへのスケーリングでは、キーの (10–1)/10 (90 パーセント) が移動されます。トラフィックのスパイクの理由からスケールアップする場合、多数のキャッシュミスが発生することは避けたいものです。多数のキャッシュミスは、トラフィックのスパイクにより既に過負荷になっているデータベースのヒットとなります。

このジレンマには、整合性のあるハッシュがソリューションとなります。整合性のあるハッシュではアルゴリズムを使用し、ノードがクラスターから追加または削除されるたびに、移動する必要のあるキーの数は約 1/n となります (n は新しいノード数)。1 ノードから 2 ノードへのスケーリングでは、キーの 1/2 (50 パーセント) が移動され、最悪のケースとなります。9 ノードから 10 ノードへのスケーリングでは、キーの 1/10 (10 パーセント) が移動されます。

ユーザーとして、複数ノードのクラスターに使用されるハッシュアルゴリズムを制御します。整合性のあるハッシュを使用するようにクライアントを設定することをお勧めします。さいわい、整合性のあるハッシュを実装する Memcached クライアントライブラリは数多くあり、ほとんどの一般的な言語で提供されています。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。

Java、PHP、または で作業している場合はNET、Amazon ElastiCache クライアントライブラリのいずれかを使用することをお勧めします。

Java を使用した整合性のあるハッシュ

ElastiCache Memcached Java クライアントは、オープンソースの spymemcached Java クライアントに基づいており、一貫したハッシュ機能が組み込まれています。ライブラリには、一貫したハッシュを実装するKetamaConnectionFactory クラスが含まれています。デフォルトでは、整合性のあるハッシュは spymemcached では無効になっています。

詳細については、「」の KetamaConnectionFactory ドキュメントを参照してくださいKetamaConnectionFactory。

Memcached PHP を使用した一貫したハッシュ

ElastiCache Memcached PHPクライアントは、組み込みの Memcached PHPライブラリを囲むラッパーです。デフォルトでは、一貫したハッシュは Memcached PHPライブラリによってオフになっています。

整合性のあるハッシュを有効にするには、以下のコードを使用します。

```
$m = new Memcached();
$m->setOption(Memcached::OPT_DISTRIBUTION, Memcached::DISTRIBUTION_CONSISTENT);
```

また、先ほどのコードに加えて、php.ini ファイルで memcached.sess_consistent_hash を有効にすることをお勧めします。

詳細については、PHPhttp://php.net/manual/en/memcached.configuration.php の Memcached のランタイム設定ドキュメントを参照してください。特に、memcached.sess_consistent_hash パラメータについて参照してください。

Memcached NETでの を使用した一貫したハッシュ

ElastiCache Memcached 。NET クライアントは Enyim Memcached の周りのラッパーです。デフォルトでは、Enyim Memcached クライアントによって、整合性のあるハッシュが有効になります。

詳細については、https://github.com/enyim/EnyimMemcached「/wiki/MemcachedClient-Configuration#user-content-memcachedlocator」のmemcached/locatorドキュメントを参照してください。

Memcached で検証されたクライアント

以下のクライアントは、Memcached でサポートされているすべてのネットワークタイプ設定で動作することが特に検証されています。

検証済みクライアント:

- AWS ElastiCache Php 用クラスタークライアント Memcached – バージョン *3.6.2
- AWS ElastiCache Cluster Client Memcached for Java – Github の最新のマスター

デュアルスタッククラスターの優先プロトコルの設定 (Memcached)

Memcached 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 Discovery パラメータは、 IPv4または のいずれかに設定できますIPv6。

IP 検出パラメータは、config get クラスター出力で使用される IP プロトコルを制御します。これにより、 ElastiCache (Memcached) クラスターの自動検出をサポートするクライアントが使用する IP プロトコルが決まります。

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。

Java の場合は getAvailableNodeEndPoints の出力をモニタリングし、PHP の場合は getServerList の出力をモニタリングします。これらの関数の出力が、更新されたプロトコルを使用するクラスター内のすべてのノードIPsについてレポートされると、変更の伝播は完了します。

Java の例:

```
MemcachedClient client = new MemcachedClient(new InetSocketAddress("xxxx", 11211));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;
```

```
do {
    nodes =
 client.getAvailableNodeEndPoints().stream().map(NodeEndPoint::getIpAddress).collect(Collectors

    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
        }));
```

PHP の例:

```
$client = new Memcached;
$client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);
$client->addServer("xxxx", 11211);

$nodes = [];
$target_ips_count = 0;
do {
    # The PHP memcached client only updates the server list if the polling interval has
 expired and a
    # command is sent
    $client->get('test');

    $nodes = $client->getServerList();

    sleep(1);
    $target_ips_count = 0;

    // For IPv4 use FILTER_FLAG_IPV4
    $target_ips_count = count(array_filter($nodes, function($node) { return
 filter_var($node["ipaddress"], FILTER_VALIDATE_IP, FILTER_FLAG_IPV6); }));

} while (count($nodes) !== $target_ips_count);
```

IP 検出が更新される前に作成された既存のクライアント接続は、引き続き古いプロトコルを使用して接続されます。クラスター検出コマンドの出力で変更が検出されると、検証されたすべてのクライアントは新しい IP プロトコルを使用してクラスターに自動的に再接続します。ただし、これはクライアントの実装によって異なります。

TLS 有効なデュアルスタック ElastiCache クラスター

TLS が ElastiCache クラスターで有効になっている場合`cluster slots`、クラスター検出関数 (、、Redis `cluster nodes`の場合は) `cluster shards`または ではなく `config get cluster` Memcached リターンホスト名IPs。その後、ホスト名は の代わりにクラスターIPsに接続 ElastiCache 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 有効なクラスターの場合、IP Discovery パラメータは優先 IP プロトコルには影響しません。代わりに、使用される IP プロトコルは、DNSホスト名を解決するときにクライアントがどの IP プロトコルを優先するかによって決まります。

Java クライアント

IPv4 と の両方をサポートする Java 環境から接続する場合IPv6、Java は下位互換性IPv6のためにデフォルトで IPv4 よりも優先されます。ただし、IP プロトコル設定は JVM引数を使用して設定できます。を優先するにはIPv4、 は をJVM承諾`-Djava.net.preferIPv4Stack=true`し、IPv6を設定することを優先します`-Djava.net.preferIPv6Stack=true`。を設定する`-Djava.net.preferIPv4Stack=true`と、 JVMはIPv6接続しなくなります。Valkey または Redis の場合OSS、これには他の非 Valkey および非 Redis OSSアプリケーションへのアプリケーションが含まれます。

ホストレベルの設定

通常、クライアントまたはクライアントのランタイムが IP プロトコル設定の設定オプションを提供していない場合、DNS解決を実行するときに IP プロトコルはホストの設定によって異なります。デフォルトでは、ほとんどのホストは IPv6 よりも優先されますIPv4が、この設定はホストレベルで設定できます。これは、 ElastiCache クラスターへのリクエストだけでなく、そのホストからのすべてのDNSリクエストに影響します。

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IPv4時に優先する設定を更新するには、クラスターを含むCIDR範囲の優先順位IPsをデフォルトのIPv6接続の優先順位よりも高く更新します。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 172.31.0.0:0/16 CIDR のサブネットにあると仮定すると、以下の設定により、クライアントはそのクラスターIPv4への接続を優先します。

```
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、指定したCIDR範囲IPv4の接続よりもIPv6接続を優先するように設定ポリシーが更新されます。例えば、クラスターが 172.31.0.0:0/16 CIDRを実行しているサブネットにあると仮定netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15すると、次の優先順位テーブルが生成され、クライアントはクラスターに接続するIPv4ときに優先されます。

```
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
```

Valkey と Redis の予約済みメモリの管理 OSS

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、Valkey と Redis は使用可能なメモリOSSを使用して、クラスターのデータが .rdb ファイルに書き込まれている間、クラスターへの書き込みオペレーションを記録します。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。次に、 ElastiCache (Redis OSS) の予約済み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。

OSSより前のバージョンの Redis を実行している場合は、Redis OSS2.8.22 以降を実行している場合よりもバックアップとフェイルオーバー用により多くのメモリを予約します。この要件は、ElastiCache (Redis OSS) がバックアッププロセスを実装する方法が異なるためです。経験則では、2.8.22 より前のバージョンの Redis OSSオーバーヘッドにはノードタイプのmaxmemory値の半分を、Redis OSSバージョン 2.8.22 以降の場合は 4 分の 1 を予約します。

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制限付近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規模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負荷の高いワークロードでは、ノードの使用可能なメモリの最大 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。

- Valkey または Redis OSS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を確認する

- 同期とバックアップの実装方法
- のデータ階層化 ElastiCache

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 は Valkey または Redis OSS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 を提供しています`reserved-memory-percent`。これらのパラメータはいずれも Valkey ディストリビューションまたは Redis OSSディストリビューションの一部ではありません。

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 Valkey または Redis OSS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 日より前に開始されたお客様の場合 – デフォルトのパラメータグループを使用して Redis OSS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 ですreserved-memory。この場合、0 バイトのメモリが予約されます。
- 2017 年 3 月 16 日以降に開始されたお客様の場合 – デフォルトのパラメータグループを使用して Valkey または Redis OSS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 ですreserved-memory-percent。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 Valkey または Redis OSSメモリ管理パラメータについて読んだ後、デフォルトではないパラメータまたはデフォルト以外の値を使用することをお勧めします。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。その後、新しい Valkey または Redis OSS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、カスタムパラメータグループを使用できます。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
- ElastiCache パラメータグループの作成

- ElastiCache パラメータグループの変更
- 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 日より前に、すべての ElastiCache (Redis OSS) 予約済みメモリ管理は パラメータを使用して行われましたreserved-memory。reserved-memory のデフォルト値は 0 です。このデフォルトは、Valkey または Redis オーバーOSSヘッドのメモリをリザーブせず、Valkey または Redis がノードのすべてのメモリをデータで消費OSSできるようにします。

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は、クラスターとクラスターのノードタイプで実行されている Valkey または Redis OSSバージョンに適した値reserved-memory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

パラメータreserved-memoryは に ElastiCache固有であり、一般的な Redis OSSディストリビューションの一部ではありません。

次の手順は、 reserved-memoryを使用して Valkey または Redis OSSクラスターのメモリを管理する方法を示しています。

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、「 ElastiCache パラメータグループの作成」を参照してください。

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis6x-m3xl \
      --description "Redis OSS 2.8.x for m3.xlarge node type" \
      --cache-parameter-group-family redis6.x
   ```

2. Valkey または Redis オーバーOSSヘッド用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 OSS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 OSSより前に Redis のバージョン

を実行しており、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、「ElastiCache パラメータグループの変更」を参照してください。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis28-m3xl \
   --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 OSS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例では、カスタムパラメータグループをredis28-m3xlすぐに使用 my-redis-clusterするようにクラスターを変更します。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-m3xl \
      --apply-immediately
   ```

   次のCLI例では、レプリケーショングループを変更my-redis-repl-grpして、カスタムパラメータグループをすぐにredis28-m3xl使用できるようにします。詳細については、「レプリケーショングループの変更」。

   ```
   aws elasticache modify-replication-group \
      --replication-group-id my-redis-repl-grp \
      --cache-parameter-group-name redis28-m3xl \
      --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、すべてのバージョンの ElastiCache (Redis) で利用可能にしましたOSS。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。

パラメータreserved-memory-percentは に固有 ElastiCache であり、一般的な Redis OSS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、「のデータ階層化 ElastiCache」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

(ElastiCache Redis OSS) クラスターのメモリを管理するreserved-memory-percentために を使用するには、次のいずれかを実行します。

- Redis OSS2.8.22 以降を実行している場合は、デフォルトのパラメータグループをクラスターに割り当てます。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 OSSより前に Redis のバージョンを実行している場合は、 reserved-memory-percentのデフォルト 25% よりも多くのメモリを予約する必要がある可能性があります。そのためには、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、「 ElastiCache パラメータグループの作成」を参照してください。

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --description "Redis OSS 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、「 ElastiCache パラメータグループの変更」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. このカスタムパラメータグループは、2.8.22 よりOSS古いバージョンの Redis を実行している Redis OSSクラスターまたはレプリケーショングループに使用します。

   次のCLI例では、Redis OSSクラスターを変更my-redis-clusterして、カスタムパラメータグループをすぐにredis28-50使用できるようにします。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例では、Redis OSSレプリケーショングループを変更my-redis-repl-grpして、カスタムパラメータグループをすぐにredis28-50使用できるようにします。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。

   ```
   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ドメモリ管理パラメータは、リザーブドメモリのreserved-memoryゼロ (0) バイトです。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ドメモリ管理パラメータはノードのメモリの reserved-memory-percent 25% が予約されています。これは、 ElastiCache (Redis OSS) 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、AWS CLI または を使用して、予約済みメモリ管理パラメータを変更できます ElastiCache API。

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、「ElastiCache パラメータグループの作成」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 とそれの値を指定します。

次のCLI例では、 reserved-memory-percentを使用して予約済みメモリを管理するredis32-cluster-onようにカスタムパラメータグループを変更します。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、「ElastiCache パラメータグループの変更」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 とそれの値を指定します。

次のCLI例では、 reserved-memoryを使用して予約済みメモリを管理するredis32-m3xlようにカスタムパラメータグループを変更します。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、「ElastiCache パラメータグループの変更」を参照してください。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-m3xl \
   --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```

Valkey および Redis OSSの独自設計クラスターを使用する際のベストプラクティス

マルチ AZ の使用、十分なメモリ、クラスターのサイズ変更、ダウンタイムの最小化はすべて、Valkey または Redis でセルフ設計のクラスターを使用する際に留意すべき有用な概念です OSS。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
- Valkey または Redis OSS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を確認する
- 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

ElastiCache Valkey または Redis がプライマリノードを置き換えるOSS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的なメンテナンスや、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の低いイベントが含まれます。

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。 ElastiCache はこれを透過的に処理するため、新しいプライマリノードを作成してプロビジョニングする必要はありません。このフェイルオーバーとレプリカの昇格により、昇格が完了したらすぐに新しいプライマリへの書き込みを再開できます。

マルチ AZ とダウンタイムの最小化の詳細については[Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS、「」を参照してください。

Valkey または Redis OSS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を確認する

Valkey 7.2 以降、および Redis OSSバージョン 2.8.22 以降のスナップショットと同期

Valkey には、スナップショットと同期のデフォルトサポートがあります。Redis 2.8.22 OSS では、同期と保存中にスワップ使用量を増やすことなく、アプリケーションの使用により多くのメモリを割り当てることができるフォークレス保存プロセスが導入されました。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 より前の Redis OSSスナップショットと同期

ElastiCache (Redis OSS) を使用する場合、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OSS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 で追加専用ファイル機能 (AOF) を有効にする場合OSS。
- 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書き込みプロセスOSSを実行するときは、プロセスのオーバーヘッドに対応するために十分なメモリが必要です。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。そのため、Redis OSSクラスターを作成するときは、十分なメモリを持つノードインスタンスタイプを選択することが重要です。

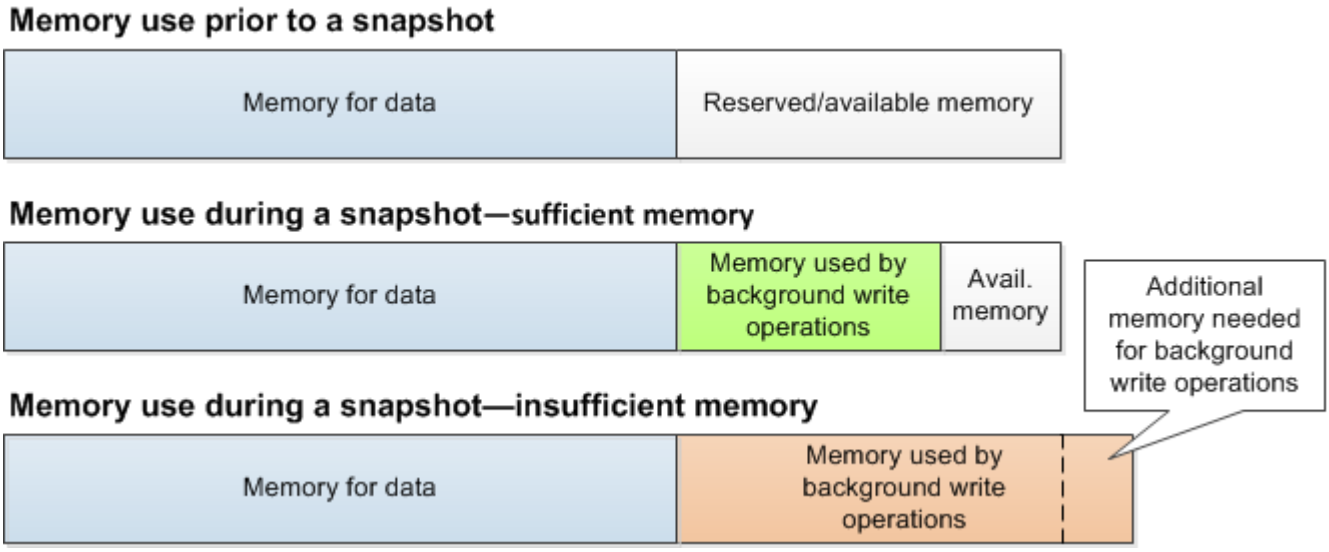
Valkey と Redis でのバックグラウンド書き込みプロセスとメモリ使用量 OSS

バックグラウンド書き込みプロセスが呼び出されるたびに、Valkey と Redis はそのプロセスをOSSフォークします (これらのエンジンはシングルスレッドであることに注意してください)。1 つのフォークは、データを Redis .rdb OSS スナップショットファイルのディスクに保持します。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使用できる限り、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。



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。

Valkey と Redis がスナップショットOSSを実行する方法の詳細については、http://valkey.io 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、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」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 や などのバックグラウンド書き込みプロセスBGREWRITEAOFが呼び出されるたびに、プロセスが失敗しないようにするには、プロセス中に書き込みオペレーションで消費されるメモリよりも多くのメモリを使用できる必要があります。最悪の場合のシナリオは、バックグラウンド書き込みオペレーション中にすべてのレコードが更新され、いくつかの新しいレコードがキャッシュに追加されることです。このため、2.8.22 より前の Redis OSSバージョンでは 50 (50%)、Valkey およびすべての Redis OSSバージョン 2.8.22 以降では 25 (25%) reserved-memory-percent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。のデフォルト値reserved-memoryは 0 です。これにより、Redis はデータを含むすべての maxmemory OSSを消費できるため、バックグラウンド書き込みプロセスなど、他の用途ではメモリが少なすぎる可能性があります。ノードインスタンスタイプごとの maxmemory 値については、「Redis OSSノードタイプ固有のパラメータ」を参照してください。

`reserved-memory` パラメータを使用して、ボックスで使用されるメモリ量を減らすこともできます。

の Valkey および 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Valkey パラメータと Redis OSS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、「 ElastiCache パラメータグループの作成」と「 ElastiCache パラメータグループの変更」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、再シャーディング中は、マルチコアインスタンスでは CPU 80% 未満、単一コアインスタンスでは 50% 未満の使用を維持することをお勧めします。アプリケーションがスケーリングの問題を監視する前に、メトリクスをモニタリング ElastiCache (Redis OSS) し、再シャーディングを開始します。追跡すると有用なメトリックスは、`CPUUtilization`、`NetworkBytesIn`、`NetworkBytesOut`、`CurrConnections`、`NewConnecti` です。
- スケーリングする前に、空きメモリが十分に確保されていることを確認する – スケーリングする場合、保持するシャードの空きメモリが、削除するシャードに使用されているメモリの 1.5 倍以上であることを確認します。
- オフピーク時にリシャーディングを開始する – このプラクティスは、リシャーディングオペレーションがクライアントのレイテンシーとスループットに与える影響を軽減するのに役立ちます。また、スロット再分散に多くのリソースを使用できるため、リシャーディングをより迅速に完了できます。
- クライアントのタイムアウト動作を確認する – オンラインクラスターのサイズ変更中に、一部のクライアントでレイテンシーが長くなる場合があります。より大きなタイムアウトでクライアントライブラリを設定すると、サーバーがより高い負荷条件でもシステムが接続する時間を与えるこ

とができます。場合によっては、サーバーへの接続を多数開く必要があります。この場合、エクスポネンシャルバックオフを追加してロジックを再接続することを検討してください。こうすると、サーバーに対して大量の新しい接続が同時に行われるのを防ぐことができます。

- 関数をすべてのシャードにロードする – クラスターをスケールアウトすると、 ElastiCache は既存のノード (ランダムに選択) の 1 つにロードされた関数を自動的に新しいノード (複数可) にレプリケートします。クラスターに Valkey 7.2 以降、または Redis OSS7.0 以降があり、アプリケーションが Functions を使用している場合は、クラスターが異なるシャード上の異なる関数を消費しないように、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する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 OSS 6 より前のBRPOPLPUSHコマンドは、移行するスロットで動作する場合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 AutoFailover が有効になっている場合、プライマリノードが変わ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。無効 AutoFailoverになっている場合、ノードのロールは変更されません。ただし、マネージドオペレーションまたはアンマネージドオペレーションのダウンタイムは、 が有効になっているAutoFailoverクラスターと比較して高くなります。

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。

Memcached のキャッシュ戦略

次のトピックでは、Memcached キャッシュの入力と維持に関する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

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、アプリケーションとアクセスするデータストア (データベース) の間にあるメモリ内キーバリューストアです。アプリケーションがデータをリクエストするたびに、まず ElastiCache キャッシュにリクエストを行います。データがキャッシュに存在し、最新である場合、はデータをアプリケーションにElastiCache 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 追加 TTL 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、新規ノードをスピンアップすると、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。存続時間 (TTL) 値 を追加することで、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"})
```

追加 TTL

遅延読み取りはデータが古くなる可能性がありますが、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。各書き込みに存続時間 (TTL) 値を追加することで、各戦略の利点を得ることができます。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。

Time to Live (TTL) は、キーの有効期限が切れるまでの秒数を指定する整数値です。Valkey または Redis OSSは、この値に秒またはミリ秒を指定できます。Memcached は、この値を秒単位で指定します。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、「Valkey および Redis OSS コマンド」または「Memcached 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、 を使用した書き込みスルーロジックの擬似コード例ですTTL。

```
// ****************************************
// 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、 を使用した遅延ロードロジックの擬似コードの例ですTTL。

```
// ****************************************
// 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
- スケーリング ElastiCache

で独自設計クラスターを管理する ElastiCache

ElastiCache には、サーバーレスキャッシュと自己設計型クラスターの 2 つのデプロイオプションがあります。それぞれに独自の機能と要件があります。

このセクションでは、独自設計のクラスターの管理に役立つトピックについて説明します。

> **Note**
>
> これらのトピックは ElastiCache Serverless には適用されません。

トピック

Auto Scaling Valkey クラスターと Redis OSSクラスター

前提条件

ElastiCache Auto Scaling は以下に限定されます。

- Valkey 7.2 以降、または Redis OSS エンジンバージョン 6.0 以降を実行している Valkey または Redis OSS (クラスターモードが有効) クラスター
- Valkey 7.2 以降、または Redis OSS エンジンバージョン 7.0.7 以降を実行しているデータ階層化 (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 - ラージ、XLarge、2XLarge
- インスタンスタイプファミリー – R7g、R6g、R6gd、R5、M7g、M6g、M5、C7gn
- の Auto Scaling ElastiCache は、グローバルデータストア、Outposts、またはローカルゾーン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。

Valkey または Redis による ElastiCache Auto Scaling によるキャパシティの自動管理 OSS

ElastiCache Valkey または Redis を使用した自動スケーリングOSSは、 ElastiCache 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。 は、Application Auto Scaling サービス

ElastiCache 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、割り当てた CloudWatch 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 自動スケーリングは、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の数を増減します。

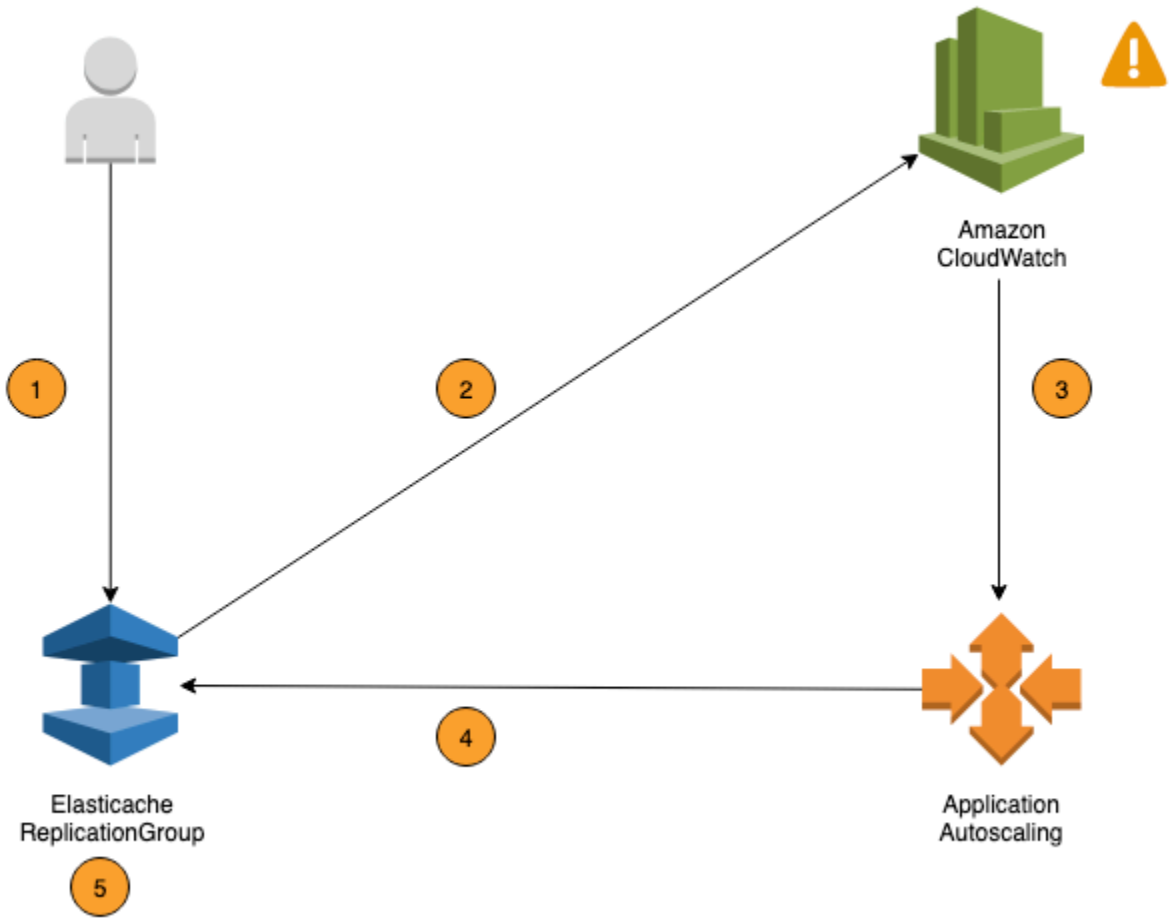
を使用して、事前定義されたメトリクスに基づいてスケーリングポリシー AWS Management Console を適用できます。`predefined metric` は列挙型で定義されるため、それをコード内に名前で指定するか、 AWS Management Consoleで使用できます。カスタムのメトリクスは、 AWS Management Consoleを使用した選択には使用できません。または、 AWS CLI または Application Auto Scaling API を使用して、事前定義されたメトリクスまたはカスタムメトリクスに基づいてスケーリングポリシーを適用することもできます。

ElastiCache Valkey または Redis では、次のディメンションのスケーリングOSSがサポートされています。

- [シャード] — 手動オンラインリシャーディングと同様に、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、 ElastiCache自動スケーリング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 Valkey または Redis OSS 自動スケーリングを使用すると、クラスター内のすべてのシャードでレプリカが均等に追加/削除されます。

ElastiCache Valkey または Redis では、次のタイプの自動スケーリングポリシーOSSがサポートされています。

- 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。
- Valkey または Redis OSS 自動スケーリング ElastiCache によるアプリケーションのスケジュールされたスケーリング – サービスが実行しているシャード/レプリカの数を、日時に基づいて増減します。



次の手順では、前の図に示すように、 ElastiCache と Valkey または Redis OSS 自動スケーリングプロセスの概要を示します。

1. レプリケーショングループの ElastiCache 自動スケーリングポリシーを作成します。
2. ElastiCache Valkey または Redis による自動スケーリングは、ユーザーに代わって CloudWatch アラームのペアOSS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。これらの CloudWatch アラームは、クラスターの実際の使用率がターゲット使用率から一定期間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. 設定されたメトリクス値が特定の期間にわたってターゲット使用率を超える (またはターゲットを下回る) 場合、 は自動スケーリング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 CloudWatch をトリガーします。
4. ElastiCache Valkey または Redis OSS 自動スケーリングを使用すると、クラスター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 Valkey または Redis では、ターゲット使用率に近づくようにクラスターのシャード/レプリカ容量を動的に増加 (または減少) して、変更リクエストOSSを処理します。

Valkey または Redis OSS Auto Scaling ElastiCache の仕組みを理解するには、 という名前のクラスターがあるとしますUsersCluster。の CloudWatch 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時にクラスターが必要とする最大シャードと、トラフィックが最低ポイントにあるときに最小シャードを決定します。また、UsersClusterクラスターのCPU使用率のターゲット値も決定します。 ElastiCache 自動スケーリングでは、ターゲット追跡アルゴリズムを使用して、 のプロビジョニングされたシャードUsersClusterが必要に応じて調整され、使用率がターゲット値に近づくようにします。

> **Note**
>
> スケーリングにはかなりの時間がかかる場合があり、シャードが再調整するには追加のクラスターリソースが必要になります。Valkey または Redis OSS Auto Scaling ElastiCache では、実際のワークロードが数分間維持されて (または抑制されて) いる場合にのみリソース設定を変更します。自動スケーリングターゲット追跡アルゴリズムは、ターゲット使用率を長期的に選択した値以下に維持することを目指します。

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 – Valkey または Redis OSS Auto Scaling ElastiCache でスケーリングするタイミングと量を決定するために使用される CloudWatch 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、現在のキャパシティが設定された最大キャパシティよりも大きい場合、ポリシーの作成 MaxCapacity 中に scaleIn に移動します。同様に、現在の容量が設定された最小容量よりも低い場合は、 scaleOut に移動します MinCapacity。

- 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 – 特定の AWS サービスにリンクされた AWS Identity and Access Management (IAM) ロール。サービスにリンクされたロールには、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれます。Valkey または Redis OSS Auto Scaling ElastiCache を使用するとAWSServiceRoleForApplicationAutoScaling_ElastiCacheRG、このロール が自動的に生成されます。
- 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

このタイプのポリシーでは、事前定義されたメトリクスまたはカスタムメトリクスとメトリクスのターゲット値は、ターゲット追跡スケーリングポリシー設定で指定されます。Valkey または Redis OSS Auto Scaling ElastiCache では、スケーリングポリシーをトリガーする CloudWatch アラームを作成して管理し、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/レプリカを追加または削除します。メトリ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リングポリシーは、変化するワークロードによるメトリ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では、CPU使用率を 70% などの指定された使用率に維持したり、近くに維持したりできます。

> **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

Valkey または Redis OSS 自動スケーリング ElastiCache を使用して、 で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。自動スケーリングで管理されるシャードの最小数を指定することもできます。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

Valkey または Redis OSS 自動スケーリング ElastiCache で 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。自動スケーリングで管理されるレプリカの最小数を指定することもできます。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 Valkey または Redis OSS 自動スケーリングポリシーを使用すると、定義され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、クラスターの容量が増加します。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 バリアントのトラフィックがゼロになると、ElastiCache Valkey または Redis OSSは自動的に指定されたインスタンスの最小数にスケールインされます。

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用の Valkey または Redis OSSクラスター ElastiCache を使用した 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用のシャード/レプリカの作成が遅くなります。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、クラスター内のシャード/レプリカの数を減らします。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、クラスター内のシャード/レプリカの数を増やします。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常に有効になっており、スケーリングポリシーは必要に応じて Valkey または Redis OSS シャード/レプリカ ElastiCache で を作成できます。

IAM Auto Scaling に必要なアクセス許可

ElastiCache Valkey または Redis OSS Auto Scaling は ElastiCache、、 CloudWatch、および Application Auto Scaling の組み合わせによって可能になりますAPIs。クラスターは ElastiCache (Redis OSS) で作成および更新され、アラームは で作成され 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IAMアクセス許可に加えて、 ElastiCache Auto Scaling 設定にアクセスするIAMユーザーは、動的スケーリングをサポートするサービスに対して適切なアクセス許可を持っている必要があります。IAM ユーザーは、次のポリシー例に示すアクションを使用するアクセス許可を持っている必要があります。

```
{
    "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 Valkey または Redis OSS 自動スケーリングサービスの には、クラスターと CloudWatch アラームを記述するアクセス許可と、ユーザーに代わって ElastiCache ターゲットキャパシティを変更するアクセス許可も必要です。クラスターの 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、フリートの現在の容量をモニタリングし、フリートの容量を変更する ElastiCache 自動スケーリングアクセス許可を付与します。サービスにリンクされ

たロールは、 ElastiCache 自動スケーリングのデフォルトロールです。詳細については、Application Auto Scaling ユーザーガイドの ElastiCache (Redis OSS) 自動スケーリング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 つだけ使用する – クラスターに CPU または データ集約型ワークロードがあるかどうかを特定し、対応する事前定義されたメトリクス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避けることをお勧めします。Valkey または Redis OSS Auto スケーリング ElastiCache を使用すると、ターゲット追跡ポリシーがスケールアウトできる状態であればスケーラブルターゲット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になっている) がスケールインできる状態である場合にのみスケールインされます。複数のポリシーによって、スケー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ベース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. ターゲット値の定義 – 4 週間にわたるクラスター使用率に対応する CloudWatch メトリクスを分析して、ターゲット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/レプリカディメンション間でワークロードを均一に分散す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**
>
> クラスターをスケールアウトすると、 ElastiCache は既存のノード (ランダムに選択) の 1 つにロードされた関数を新しいノード (複数可) に自動的にレプリケートします。クラスターに Valkey または Redis OSS7.0 以降があり、アプリケーションが Functions を使用している場合は、クラスターがさまざまなシャードで異なる関数を消費しないように、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する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

ElastiCacheでは AutoScaling 、Valkey または Redis OSS エンジンで追跡ポリシーとスケジュールされたポリシーを使用できます。

以下は、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 と を使用してそれらを適用する方法を示していますAPIs。

トピック

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 Valkey または Redis OSS Auto Scaling では、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で、またはその近くで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
ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage	DatabaseCapacityUsageCountedForEvictPercentage	Valkey または Redis OSS レプリケーショングループのメトリクス	なし
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage	Valkey または Redis OSS レプリケーショングループのメトリクス	R6gd

データ階層インスタンスタイプは、メモリと の両方にデータを保存するためElastiCacheDatabaseMemoryUsageCountedForEvictPercentage、 を使用できません

SSD。データ階層型インスタンスの想定されるユースケースは、メモリ使用率を 100% にしSSD、必要に応じて補充することです。

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 Valkey または Redis を使用すると、クラスターシャードは、ターゲットからのばらつき率と現在のシャードの 20% という 2 つの数値のうち大きい方に等しい数だけOSSスケールアウトされます。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% 未満でない限り、自動スケールイン ElastiCache は行われません。

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% 違反した場合、 ElastiCache Valkey または Redis は 30% OSSスケールアウトし、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% 違反した場合、 ElastiCache Valkey または Redis はデフォルトで最低 20% OSSスケールアウトし、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は、ターゲット値を 60% を選択した場合、メトリクスが 45% 以下 (ターゲット 60% を 25% 下回る) になるまで、 ElastiCache Valkey または Redis は自動スケールインOSS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ると、ターゲット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はできません。Valkey または Redis ElastiCache を使用すると、クラスター内の既存のシャードのターゲットから少なくとも 20% の偏差だけシャードをOSSスケールアウトできます。
- 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。不十分なデータの使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、Valkey または Redis OSS Auto Scaling ElastiCache では、追加または削除する容量を決定するときに、常に切り上げまたは切り下げによって保守的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。
- Valkey クラスターまたは Redis OSSクラスター ElastiCache を使用する には、それぞれが異なるメトリクスを使用するという条件で、複数のターゲット追跡スケーリングポリシーを設定できます。 ElastiCache (Redis OSS) Auto Scaling の目的は、常に可用性に優先順位を付けることです。そのため、ターゲット追跡ポリシーのスケールアウトまたはスケールインの準備が整っているかどうかによって動作が異なります。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る場合にのみスケールインされます。
- Valkey または Redis OSS Auto Scaling ElastiCache がターゲット追跡スケーリングポリシーに対して管理する CloudWatch アラームを編集または削除しないでください。 ElastiCache Auto Scaling は、スケーリングポリシーを削除するとアラームを自動的に削除します。
- ElastiCache Auto Scaling では、クラスターシャードを手動で変更することはできません。これらの手動調整は、スケーリングポリシーにアタッチされている既存の CloudWatch アラームには影響しませんが、これらの CloudWatch アラームをトリガーするメトリクスに影響を与え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のために不要になったスケーリング
  - アラームに影響しているすべてのシャードAVGに集約されているため、必要に応じてスケーリングを行わない。
- ElastiCache クラスターあたりのノードには、引き続き Valkey または Redis のOSSデフォルト制限が適用されます。したがって、Auto Scaling を選択するときに、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十分な ENIs (Elastic Network Interfaces) VPCが に用意されてい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。
- から利用できる容量が十分でない場合EC2、 ElastiCache Auto Scaling は容量が使用可能になるまでスケーリングされず、遅延します。
- ElastiCache (Redis OSS) スケールイン中の Auto Scaling では、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は削除されません。

- 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

を使用してスケーリングポリシーを追加できます AWS Management Console。

Valkey または Redis OSSクラスター ElastiCache を持つ に Auto Scaling ポリシーを追加するには

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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]。キャパシティメトリクスには、データ階層インスタンスのメモリとSSD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。自動スケーリングでは、 ElastiCache シャード全体で選択したターゲットメトリクスに対してこの値が維持されます。
   - プライマリCPU使用率: は、プライマリノードの`EngineCPUUtilization`メトリクスのターゲット値を維持します。
   - 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. 最小容量 には、 ElastiCache 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. 最大容量 には、 ElastiCache 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

Valkey または Redis OSSクラスターで ElastiCache で Auto Scaling を使用する前に、クラスターを ElastiCache 自動スケーリングに登録します。これにより、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します。 ElastiCache 自動スケーリングは、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に沿ってクラスターを動的にスケーリングします。

の使用 AWS CLI

ElastiCache を Valkey または Redis OSSクラスターに登録するには、次のパラメータで register-scalable-target コマンドを使用します。

- `--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。このパラメータでは、リソースタイプは `ReplicationGroup`で、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- `--scalable-dimension` – この値を `elasticache:replication-group:NodeGroups` に設定します。
- `--max-capacity` – ElastiCache 自動スケーリングで管理され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 自動スケーリングによって管理され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

次の例では、 という名前の Valkey または Redis OSSクラスター ElastiCache に を登録しますmyscalablecluster。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 ^
```

の使用 API

ElastiCache クラスターを登録するには、次のパラメータで register-scalable-target コマンドを使用します。

- ServiceNamespace – この値を 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、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- ScalableDimension – この値を にelasticache:replication-group:NodeGroups設定します。
- MinCapacity – ElastiCache 自動スケーリングによって管理され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 自動スケーリングで管理され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

次の例では、Application Auto Scaling ElastiCache で という名前の Valkey または Redis OSSクラスターmyscalableclusterに を登録しますAPI。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ブロックによって表されます。スケーリングポリシー設定は、テキストファイルにJSONブロックとして保存できます。 AWS CLI または Application Auto Scaling を呼び出すときに、そのテキストファイルを使用しますAPI。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TargetTrackingScalingPolicyConfigurationの「」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

事前定義メトリクスの使用

事前定義されたメトリクスを使用することで、 ElastiCache (Redis OSS) Auto Scaling のターゲット追跡で動作する Valkey または Redis OSSクラスター ElastiCache を持つ 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。

現在、 は NodeGroup Auto Scaling で以下の事前定義されたメトリクス ElastiCache をサポートしています。

- ElastiCachePrimaryEngineCPUUtilization – クラスター内のすべてのプライマリノードにわたる CloudWatchの`EngineCPUUtilization`メトリクスの平均値。
- ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage – クラスター内のすべてのプライマリノード CloudWatch にわたる の`DatabaseMemoryUsageCountedForEvictPercentage`メトリクスの平均値。
- ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage – クラスター内のすべてのプライマリノード CloudWatch にわたる の`ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage`メトリクスの平均値。

`EngineCPUUtilization` と `DatabaseMemoryUsageCountedForEvictPercentage`、および `DatabaseCapacityUsageCountedForEvictPercentage` メトリクスの詳細については、「CloudWatch メトリクスの使用のモニタリング」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、事前定義されたメトリクス`PredefinedMetricSpecification`の と TargetValue 、そのメトリクスのターゲット値の を含める必要があります。

Example

次の例では、Valkey または Redis OSSクラスター ElastiCache を使用する 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平均CPU使用率が 40% に基づいてクラスターを調整します。

```
{
```

```
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    }
}
```

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メトリクスに基づいて、カスタム ElastiCache メトリクスを定義できます。すべての ElastiCache メトリクスがターゲット追跡に機能す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。クラスター内のシャード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。この設定では、カスタムメトリクスは、 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 50% に基づいて ElastiCache (Redis OSS) クラスターを調整します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TargetTrackingScalingPolicyConfigurationの「」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クスを使用して、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 ElastiCache (Redis OSS) クラスターを調整します。この構成では、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クラスター内で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TargetTrackingScalingPolicyConfigurationの「」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クスは、そのOSSクラスター内のすべてのプライマリノードの平均CPU使用率が 40% に基づいて、Valkey または Redis クラスター ElastiCache で 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 ElastiCache に Valkey または Redis OSS 自動スケーリングに登録し、スケーリングポリシーを定義した後、スケーリングポリシーを登録済みのクラスターに適用します。(ElastiCache Redis OSS) 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 Application Auto Scaling を使用できますAPI。

を使用したスケーリングポリシーの適用 AWS CLI

Valkey または Redis OSSクラスター ElastiCache を使用して にスケーリングポリシーを適用するには、次のパラメータで 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 – 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、一意の識別子はクラスターの名前です。例えば、 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 – 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。

次の例では、 という名前のターゲット追跡スケーリングポリシー`myscalablepolicy`を、ElastiCache 自動スケーリング ElastiCache で という名前の Valkey または Redis OSSクラスター`myscalablecluster`を持つ に適用します。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 \
    --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
```

を使用したスケーリングポリシーの適用 API

Valkey または Redis OSSクラスター ElastiCache を使用して にスケーリングポリシーを適用するには、次のパラメータで PutScalingPolicy 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 – 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、一意の識別子はクラスターの名前です。例えば、 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 – 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。

次の例では、 という名前のターゲット追跡スケーリングポリシーmyscalablepolicyを、ElastiCache 自動スケーリング ElastiCache で という名前の Valkey または Redis OSSクラスターmyscalableclusterを持つ に適用しま

す。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

スケーリングポリシーは、 AWS Management Console、、 AWS CLIまたは Application Auto Scaling を使用して編集できますAPI。

を使用したスケーリングポリシーの編集 AWS Management Console

Valkey または Redis OSSクラスター ElastiCache を持つ の Auto Scaling ポリシーを編集するには

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、適切なエンジンを選択します。
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 を選択します。

AWS CLI および を使用したスケーリングポリシーの編集 API

AWS CLI または Application Auto Scaling 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APIを編集できます。

- を使用する場合は AWS CLI、 `--policy-name`パラメータで編集するポリシーの名前を指定します。変更するパラメータの新しい値を指定します。
- Application Auto Scaling を使用する場合はAPI、 `PolicyName`パラメータで編集するポリシーの名前を指定します。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

スケーリングポリシーは、 AWS Management Console、、 AWS CLIまたは Application Auto Scaling を使用して削除できますAPI。

を使用したスケーリングポリシーの削除 AWS Management Console

ElastiCache (Redis OSS) クラスターの Auto Scaling ポリシーを削除するには

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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。

を使用したスケーリングポリシーの削除 AWS CLI

Valkey または Redis OSSクラスター ElastiCache を使用して のスケーリングポリシーを削除するには、次のパラメータで 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 – 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、一意の識別子はクラスターの名前です。例えば、 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。

次の例では、 という名前のクラスターmyscalablepolicyから という名前のターゲット追跡スケーリングポリシーを削除しますmyscalablecluster。

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
```

を使用したスケーリングポリシーの削除 API

Valkey または Redis OSSクラスター ElastiCache を使用して のスケーリングポリシーを削除するには、次のパラメータで 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 – 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、一意の識別子はクラスターの名前です。例えば、 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。

次の例では、 という名前のクラスターmyscalablepolicyから という名前のターゲット追跡スケーリングポリシーを削除しますmyscalablecluster。

```
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 AWS CloudFormation に使用する

このスニペットは、ターゲット追跡ポリシーを作成し、AWS::ElastiCache::ReplicationGroup リソースを使用して AWS::ApplicationAutoScaling::ScalableTarget 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。スケジュールされたスケーリングを使用するには、Valkey または Redis ElastiCache に特定の時間にスケーリングアクティビティを実行するOSSように指示するスケジュールされたアクションを作成します。スケジュールされたアクションを作成するときは、既存の ElastiCache (Redis OSS) クラスター、スケーリングアクティビティが発生するタイミング、最小容量、最大容量を指定します。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 (Redis OSS) クラスターに対してのみ、スケジュールされたアクションを作成できます。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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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 application-autoscaling を使用しますAPIs。

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

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

このスニペットは、ターゲット追跡ポリシーを作成し、AWS:::ElastiCache::ReplicationGroup リソースを使用して AWS::ApplicationAutoScaling::ScalableTarget 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

ElastiCache レプリケーショングループは、単一の論理ノードとして機能するように 1 つ以上のキャッシュを設定できます。

以下は、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 と を使用してそれらを適用する方法を示していますAPIs。

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 Valkey または Redis では、スケーリングポリシーをトリガーする CloudWatch アラーム OSS AutoScaling 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`: AVG エンジンCPU使用率のしきい値は、ElastiCache を使用して自動スケーリングオペレーションをトリガーするすべてのレプリカに集約されます。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`メトリクスがターゲット設定以上であることを検出すると、シャード全体のレプリカが自動的に増加します。Valkey または Redis

ElastiCache を使用すると、クラスターレプリカはターゲットと 1 つのレプリカからのバリエーションの割合という 2 つの数値のうち大きい方に等しい数OSSだけスケールアウトされます。スケールインの場合、 ElastiCache Valkey または Redis では、全体的なメトリクス値が定義されたターゲットの 75% 未満でない限り、自動スケールインOSSは行われません。

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:

Target が 30% 違反した場合、 ElastiCache Valkey または Redis 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)) OSS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 つのシャードが生成されます。

スケールインの例では、ターゲット値を 60% を選択した場合、メトリクスが 45% 以下 (ターゲット 60% を 25% 下回る) になるまで、 ElastiCache Valkey または Redis は自動スケールインOSS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ると、ターゲット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はできません。Valkey または Redis ElastiCache では、クラスター内のすべてのシャードにわたって既存のレプリカの最大 (ターゲットから四捨五入された偏差率、デフォルト 1) だけレプリカをOSS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、Valkey または Redis OSS Auto Scaling ElastiCache では、追加または削除する容量を決定するときに、常に切り上げまたは切り下げによって保守的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。
- Valkey クラスターまたは Redis OSSクラスター ElastiCache を使用する の複数のターゲット追跡スケーリングポリシーを設定できます。ただし、それぞれが異なるメトリクスを使用する場合に限

ります。Auto Scaling の目的は、常に可用性に優先順位を付けることです。そのため、ターゲット追跡ポリシーのスケールアウトまたはスケールインの準備が整っているかどうかによって動作が異なります。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る場合にのみスケールインされます。

- Valkey または Redis OSS Auto Scaling ElastiCache がターゲット追跡スケーリングポリシーで管理する CloudWatch アラームを編集または削除しないでください。Auto Scaling は、スケーリングポリシーを削除するか、クラスターを削除すると、アラームを自動的に削除します。
- ElastiCache Valkey または Redis OSS Auto Scaling を使用しても、シャード間でレプリカを手動で変更することはできません。これらの手動調整は、スケーリングポリシーにアタッチされている既存の CloudWatch アラームには影響しませんが、これらの CloudWatch アラームをトリガーするメトリクスに影響を与え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のAVGメトリクスにわたって定義されます。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のために不要になったスケーリング
  - アラームが侵害されないように影響するすべてのシャードAVGに集約されるため、必要に応じてスケーリングしない。
- ElastiCache クラスターあたりのノードに対する Valkey または Redis のOSS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 を選択するときに、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十分な ENIs (Elastic Network Interfaces) VPCが に用意されてい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。
- から十分な容量がない場合EC2、 ElastiCache Valkey または Redis OSS Auto Scaling では、容量が使用可能になるまでスケールアウトされません。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してもスケールアウトされません。
- ElastiCache Valkey または Redis OSS Auto Scaling では、クラスターが 25% `ReservedMemoryPercent`未満のレプリカ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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

スケーリングポリシーの追加

を使用してスケーリングポリシーを追加できます AWS Management Console。

を使用したスケーリングポリシーの追加 AWS Management Console

Valkey または Redis ElastiCache を使用して自動スケーリングポリシーを に追加するには OSS

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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. [Scaling policies] (スケーリングポリシー) の下で、[Add dynamic scaling] (動的なスケーリングを追加) 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. [スケーラブルディメンション] では、ダイアログボックスから [レプリカ] を選択します。
9. ターゲット値に、 ElastiCache レプリカで維持するCPU使用率の平均パーセンテージを入力します。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 と Valkey または Redis OSS 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、 を維持するのに必要な Valkey または Redis OSS Auto Scaling ポリシー ElastiCache を持つ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. [Create] (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 Application Auto Scaling を使用できますAPI。最初のステップは、 ElastiCache を Valkey または Redis OSSレプリケーショングループに Auto Scaling に登録することです。

クラスターで ElastiCache 自動スケーリングを使用する前に、クラスターを ElastiCache に Valkey または Redis OSS 自動スケーリングに登録する必要があります。これにより、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します。Valkey または Redis OSS 自動スケーリング ElastiCache を使用すると、シャードあたりのクラスターレプリカの数を表すelasticache:replication-group:Replicasスケーラブルディメンションに沿ってクラスターが動的にスケーリングされます。

の使用 CLI

ElastiCache クラスターを登録するには、次のパラメータで register-scalable-target コマンドを使用します。

- —service-namespace – この値は elasticache に設定します。
- --resource-id – 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、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- —scalle-dimension — この値は elasticache:replication-group:Replicas に設定します。
- --min-capacity – Valkey または Redis OSS 自動スケーリング ElastiCache で が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。
- --max-capacity – Valkey または Redis OSS 自動スケーリング ElastiCache で が管理す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、 という名前の Valkey または Redis OSSクラスター ElastiCache に を登録します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 ^
```

の使用 API

ElastiCache クラスターを登録するには、次のパラメータで register-scalable-target コマンドを使用します。

- ServiceNamespace – この値を 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、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- ScalableDimension – この値を に設定しますelasticache:replication-group:Replicas。
- MinCapacity – Valkey または Redis OSS 自動スケーリング ElastiCache で によって管理され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 – Valkey または Redis OSS 自動スケーリング ElastiCache で が管理す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、 という名前のクラスターmyscalableclusterを Application Auto Scaling に登録しますAPI。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の設定は、メトリクスとターゲット値が定義されているJSONブロックによって表されます。スケーリングポリシー設定は、テキストファイルにJSONブロックとして保存できます。 AWS CLI または Application Auto Scaling を呼び出すときに、そのテキストファイルを使用しますAPI。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TargetTrackingScalingPolicyConfiguration」の「」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 AWS CloudFormation に使用する
- 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の設定は、メトリクスとターゲット値が定義されているJSONブロックによって表されます。スケーリングポリシー設定は、テキストファイルにJSONブロックとして保存できます。 AWS CLI または Application Auto Scaling を呼び出すときに、そのテキストファイルを使用しますAPI。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TargetTrackingScalingPolicyConfiguration」の「」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
- Valkey または Redis OSSクラスター ElastiCache を使用した 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

事前定義されたメトリクスを使用することで、Valkey または Redis Auto Scaling ElastiCache で のターゲット追跡を操作する Valkey または Redis OSSクラスター ElastiCache を持つ のターゲット追跡OSSスケーリングポリシーをすばやく定義できます。現在、 は、 ElastiCache Replicas Auto Scaling で次の事前定義されたメトリクス ElastiCache をサポートしています。

`ElastiCacheReplicaEngineCPUUtilization` – クラスター内のすべてのレプリカ CloudWatch にわたる の E ngineCPUUtilization メトリクスの平均値。必須 ReplicationGroupId およびロールレプリカの集計メトリクス値は、 の CloudWatch にあります ElastiCache `ReplicationGroupId`, `Role`。

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 Valkey または Redis メトリクス ElastiCache を持つ に基づいて、カスタムOSSメトリクスを定義できます。すべての ElastiCache メトリクスがターゲット追跡に機能す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は、 という名前のクラスター内のすべてのレプリカの平均CPU使用率が 50% に基づいてクラスターを調整しますmy-db-cluster。

```
{"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 の詳細についてはScaleOutCooldown、Application Auto Scaling APIリファレンスTargetTrackingScalingPolicyConfigurationの「」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、ElastiCacheReplicaEngineCPUUtilization事前定義されたメトリクスを使用して、そのクラスター内のすべてのレプリカの平均CPU使用率が 40% に基づいてクラスターを調整します。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": "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で、ターゲット追跡スケーリングポリシー設定を Valkey または Redis OSSクラスター ElastiCache で でスケーリングし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。の詳細についてはDisableScaleIn、「Application Auto Scaling APIリファレンスTargetTrackingScalingPolicyConfiguration」の「」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事前定義されたメトリクスは、そのクラスター内のすべてのレプリカの平均CPU使用率が 40% に基づいて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
}
```

Valkey または Redis OSSクラスター ElastiCache を使用した へのスケーリングポリシーの適用

Valkey または Redis OSS 自動スケーリング ElastiCache でクラスターを に登録し、スケーリングポリシーを定義したら、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。Valkey または Redis OSSクラスター ElastiCache を持つ にスケーリングポリシーを適用するには、 AWS CLI または Application Auto Scaling を使用できますAPI。

の使用 AWS CLI

Valkey または Redis OSSクラスター ElastiCache を使用して にスケーリングポリシーを適用するには、次のパラメータで 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 – 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、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- —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 – 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、 という名前のターゲット追跡スケーリングポリシーmyscalablepolicyを、Valkey または Redis OSS 自動スケーリング ElastiCache を使用する myscalablecluster という名前のクラスターに適用します。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 \
    --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
```

の使用 API

Application Auto Scaling ElastiCache を使用して Valkey または Redis OSSクラスターで にスケーリングポリシーを適用するにはAPI、次のパラメータを使用して PutScalingPolicy Application 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 – この値を に設定しますTargetTrackingScaling。
- ResourceID – 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 (Redis OSS) クラスターの名前です。例えば、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。
- TargetTrackingScalingPolicyConfiguration – 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、 という名前のターゲット追跡スケーリングポリシーscalablepolicyを、Valkey または Redis OSS 自動スケーリング ElastiCache を使用する myscalablecluster という名前のクラスターに適用します。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

スケーリングポリシーは、 AWS Management Console、、 AWS CLIまたは Application Auto Scaling を使用して編集できますAPI。

を使用したスケーリングポリシーの編集 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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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 を選択します。
8. ポリシーを変更します。
9. Modify を選択します。

AWS CLI または Application Auto Scaling を使用したスケーリングポリシーの編集 API

AWS CLI または Application Auto Scaling 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APIを編集できます。

- Application Auto Scaling を使用する場合はAPI、 `PolicyName`パラメータで編集するポリシーの名前を指定します。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、「Valkey または Redis OSSクラスター ElastiCache を使用した 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 Management Console、 AWS CLI または Application Auto Scaling を使用してスケーリングポリシーを削除できます。 API

を使用したスケーリングポリシーの削除 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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインで、Valkey または Redis OSS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 Application Auto Scaling を使用したスケーリングポリシーの削除 API

AWS CLI または Application Auto Scaling を使用してAPI、 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

Valkey または Redis OSSクラスター ElastiCache を使用して からスケーリングポリシーを削除するには、次のパラメータで delete-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。
- --resource-id – 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、一意の識別子はクラスターの名前です。例えば、 ですreplication-group/myscalablecluster。
- —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。

Example

次の例では、 という名前のターゲット追跡スケーリングポリシーを ELCという名前のクラスターmyscalablepolicyから削除しますmyscalablecluster。

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 \
```

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

Valkey または Redis OSSクラスター ElastiCache を使用して からスケーリングポリシーを削除するには、次のパラメータで DeleteScalingPolicy Application 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 – 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、一意の識別子はクラスターの名前です。例えば、 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 myscalableclusterで という名前のクラスターmyscalablepolicyから という名前のターゲット追跡スケーリングポリシーを削除しますAPI。

```
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 AWS CloudFormation に使用する

このスニペットは、スケジュールされたアクションを作成し、AWS::ElastiCache:ReplicationGroupリソースを使用して AWS::ApplicationAutoScaling::ScalableTarget 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。スケジュールされたスケーリングを使用するには、Valkey または Redis ElastiCache に特定の時間にスケーリングアクティビティを実行するOSSように指示するスケジュールされたアクションを作成します。スケジュールされたアクションを作成するときは、スケーリングアクティビティが発生するタイミング、最小容量、最大容量を Valkey または Redis OSSクラスター ElastiCache で指定します。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。

のスケジュールされたアクションは、既に存在する ElastiCache Valkey または Redis OSSクラスターでのみ作成できます。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 application-autoscaling を使用しますAPIs。

- put-scheduled-action
- describe-scheduled-actions

- delete-scheduled-action

AWS CloudFormation を使用して Auto Scaling ポリシーを作成する

このスニペットは、スケジュールされたアクションを作成し、AWS:::ElastiCache:ReplicationGroupリソースを使用して AWS::ApplicationAutoScaling::ScalableTarget 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

Valkey と Redis OSS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インメモリデータベースです。 ElastiCache Valkey クラスターと Redis OSSクラスターは、データを複数のノードに分割できる分散実装です。 ElastiCache (Redis OSS) クラスターには、クラスターモードが有効 () とクラスターモードが無効 (CME) の 2 つの操作モードがありますCMD。ではCME、Valkey と Redis OSS エンジンは複数のシャードとノードを持つ分散データベースとして機能し、 ではCMD、Valkey と Redis は単一のノードとして機能しOSSます。

CMD から に移行する前にCME、次の条件を満たす必要があります。

> **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 Valkey または Redis OSS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。
- 移行に必要な最小エンジンバージョンは Valkey 7.2 以降、または Redis 7.0 OSS 以降です。

から に移行するにはCMDCME、クラスターモード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 (Redis OSS) は、クラスターが有効な設定で対応するデフォルトのパラメータグループを自動的に選択します。クラスター対応パラメータ値は、CMDクラスターnoに対して 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 ElastiCache パラメータの設定」を参照してください

1. 準備 – テストCMEクラスターを作成し、スタックが機能する準備ができていることを確認します。 ElastiCache （ Redis OSS) には、準備状況を確認する方法はありません。詳細については、「Valkey または Redis 用のクラスターの作成 OSS」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する – このモードでは、単一のシャードがデプロイされ、 ElastiCache (Redis OSS) は単一のノードとしてだけでなく、単一の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、ア

プリケーションを再設定して、Valkey または Redis OSS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必要があります。Valkey または Redis OSS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パラメータを定義する場合、パラメータ (を除くcacheParameterGroupName) を変更することはできません。

a. を使用して AWS Management Console、「」を参照レプリケーショングループの変更し、クラスターモードを互換に設定します。

b. を使用してAPI、「」を参照ModifyReplicationGroupし、 `ClusterMode`パラメータを に更新しますcompatible。

c. を使用して AWS CLI、「」を参照modify-replication-groupし、 `cluster-mode`パラメータを に更新しますcompatible。

Valkey または Redis OSS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、DescribeReplicationGroupsAPIは ElastiCache (Redis OSS) 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、「での接続エンドポイントの検索 ElastiCache」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、単一つ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。

   開始する前に、Valkey または Redis OSSクライアントがクラスタープロトコルを使用して に移行し、クラスターの設定エンドポイントが使用されていないことを確認してください。

   a. を使用して AWS Management Console、「」を参照レプリケーショングループの変更し、クラスターモードを Enabled に設定します。

b. を使用してAPI、「」を参照ModifyReplicationGroupし、 `ClusterMode`パラメータを に更新しますenabled。

c. を使用して AWS CLI、「」を参照modify-replication-groupし、 `cluster-mode`パラメータを に更新しますenabled。

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 Valkey または Redis OSS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、元の設定を保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. を使用して AWS Management Console、「」を参照レプリケーショングループの変更し、クラスターモードを無効に設定します。
2. を使用してAPI、「」を参照ModifyReplicationGroupし、 `ClusterMode`パラメータを に更新しますdisabled。
3. を使用して AWS CLI、「」を参照modify-replication-groupし、 `cluster-mode`パラメータを に更新しますdisabled。

クラスターモードを無効に変更すると、 DescribeReplicationGroupsAPIはクラスターモードパラメータを としてdisabled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。

グローバルデータストア機能を使用すると、 AWS リージョン間でフルマネージド、高速、信頼性、安全性の高い Valkey または Redis OSSクラスター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、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実現できます。

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

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 リージョンにある必要があります。

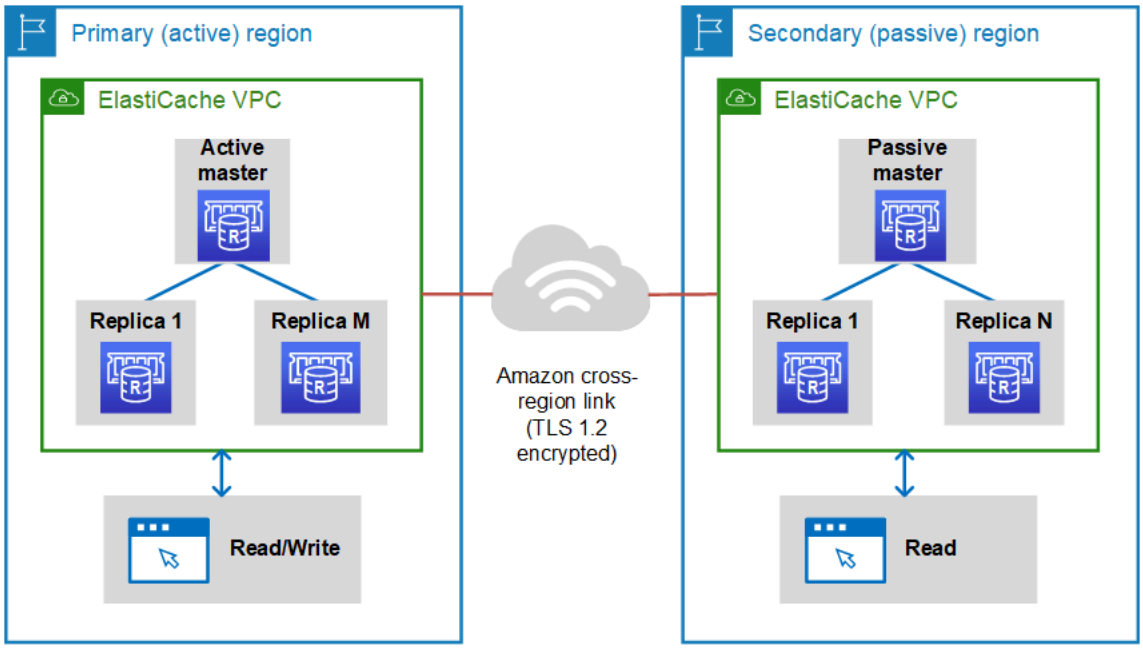
Valkey または Redis ElastiCache を使用して でグローバルデータストアを作成するとOSS、プライマリクラスターからセカンダリクラスターにデータが自動的にレプリケートされます。Valkey または Redis OSSデータをレプリケートする AWS リージョンを選択し、その AWS リージョンにセカンダリクラスターを作成します。 ElastiCache 次に、2 つのクラスター間のデータの自動非同期レプリケーションを設定および管理します。

Valkey または Redis のグローバルデータストアを使用すると、次の利点OSS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、それらの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同じである必要があります (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、サイズが 以上のインスタンスでサポートされています。

- プライマリクラスターのレプリケーションを 1 つの AWS リージョンから、最大 2 つの他の 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。

- グローバルデータストアは、VPCクラスターでのみ使用できます。詳細については、「Amazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC」を参照してください。EC2-Classic を使用する場合、グローバルデータストアはサポートされていません。詳細については、「Amazon ユーザーガイド」のEC2「-Classic」を参照してください。 EC2

  > **Note**
  >
  > 現時点では、でのローカルゾーンの使用 ElastiCache 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、「スケーリング ElastiCache」を参照してください。
- グローバルデータストアは、保管時の暗号化 、転送中の暗号化 、および をサポートしていますAUTH。
- グローバルデータストアは、Internet Protocol バージョン 6 () をサポートしていませんIPv6。

- 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公開されたイベントは、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け取ります。
>   - 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 Valkey 7.2 以降、または Redis 5.0.6 OSS 以降である必要があります。
2. Valkey 7.2 以降、または 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 OSS 2 つのセカンダリクラスターを追加します。

以下の手順では、Valkey または Redis のグローバルデータストアを作成しOSS、 ElastiCache コンソールを使用して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**
>
> 既存のクラスターは、Valkey 7.2 以降または Redis 5.0.6 OSS 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、以下を実行します。
   - 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的に生成し、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、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。Valkey 7.2 以降および Redis OSS 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、次のいずれかのアクセスコントロールオプションを指定するように求められます。
  - [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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。
  - AUTH デフォルトユーザー – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

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)
- 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. [Create] (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。

   ページが更新され、クラスターがグローバルデータストアの一部であるかどうかが示されます。これには、以下が含まれます。

   - [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、「Valkey パラメータと Redis OSSパラメータ」および「 ElastiCache パラメータグループの作成」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、「のデータ階層化 ElastiCache」を参照してください。

e. Valkey または Redis OSS (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. Valkey または Redis OSS (クラスターモードが有効) クラスターを作成する場合：

i. シャードの数 で、この Valkey または 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 OSS (パーティション/ノードグループ) の数を選択します。

Valkey または Redis の一部のバージョン OSS (クラスターモードが有効) では、クラスター内のシャードの数を動的に変更できます。

- Redis OSS3.2.10 以降 – クラスターが Redis OSS3.2.10 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の数を動的に変更できます。詳細については、「Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)」を参照してください。
- その他の Redis OSSバージョン – クラスターがバージョン 3.2.10 OSSより前のバージョンの Redis を実行している場合は、別のアプローチがあります。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。

Valkey または Redis OSS (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) あたりのレプリカの数が増減する場合は、新しい数のレプリカ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、「チュートリアル: 外部で作成されたバックアップを使用して、新しい独自設計クラスターをシードする」を参照してください。

6. サブネットグループ設定 では、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の作成を選択できます。ではIPv6、 IPv6 CIDR ブロックを使用して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、 IPv6または のいずれかの Discovery IP タイプを選択する必要がありますIPv4。

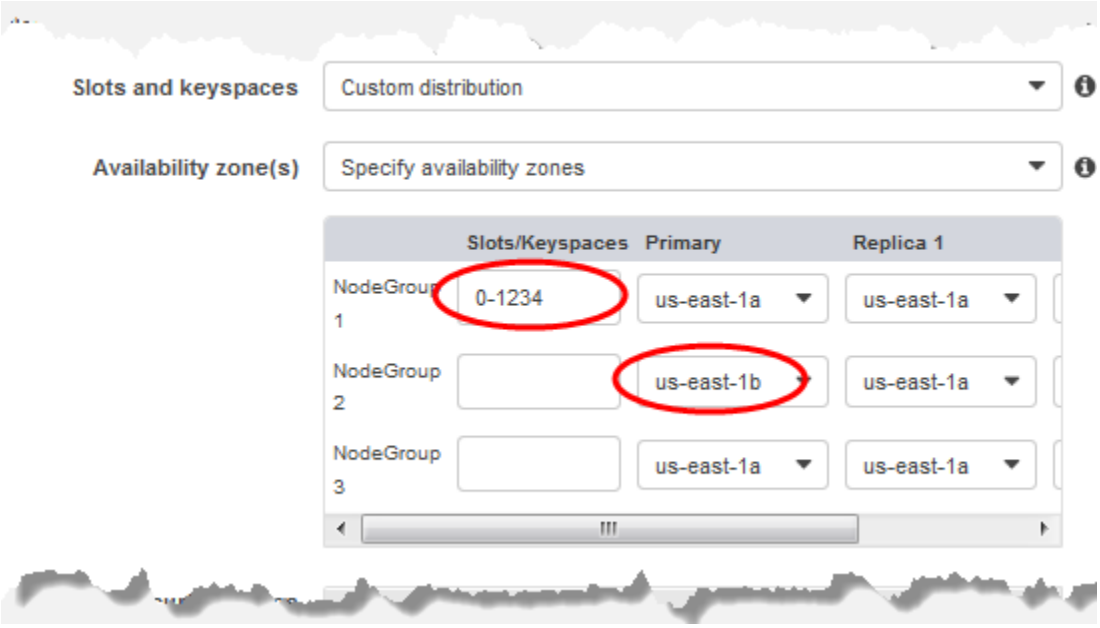
   詳細については、「 でサブネットを作成するVPC」を参照してください。

7. [Availability zone placements] (アベイラビリティーゾーンの配置) には 2 つのオプションがあります。

   - 設定なし – アベイラビリティーゾーン ElastiCache を選択します。
   - [アベイラビリティーゾーンの指定] – 各クラスターに対するアベイラビリティーゾーンを指定します。

     アベイラビリティー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ーゾーンを選択します。

   詳細については、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」を参照してください。



キースペースとアベイラビリティーゾーンの指定

8. [Next] (次へ) を選択します。
9. 高度な Valkey と Redis OSSの設定

   - [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、「 の AWS カスタマーマネージドキーの使用KMS」を参照してください。

     - [転送時の暗号化] – 転送中のデータの暗号化を有効にします。詳細については、「転送中の暗号化」を参照してください。Valkey 7.2 以降および Redis OSS 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、次のいずれかのアクセスコントロールオプションを指定するように求められます。
       - 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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。
- AUTH デフォルトユーザー – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

- AUTH – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

> **Note**
>
> OSS バージョン 3.2.10 を除く 3.2.6 以降の Redis バージョンでは、 が唯一のオプションAUTHです。

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。のデフォルトのセキュリティグループを使用するVPCか、新しいセキュリティグループを作成できます。

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

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。

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

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、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時間、通常は 1 時間です。ElastiCache は、メンテナンスウィンドウの日時を選択する (設定なし) ことも、日、時刻、期間を自分で選択することもできます (メンテナンスウィンドウを指定) 。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。すべての時間は UCT 時間です。

    詳細については、「 ElastiCache クラスター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]:

- ログ形式 で、テキストまたは を選択しますJSON。
- 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
- ログの送信先 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、タグの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、クラスターがある 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的に生成し、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**
>
> Customer Managed 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。Valkey 7.2 以降および Redis OSS エンジンバージョン 6.4 以降では、転送中の暗号化を有効にすると、次のいずれかのアクセスコントロールオプションを指定するように求められます。
  - 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、「コンソールと を使用したユーザーグループの管理 CLI」を参照してください。
  - AUTH デフォルトユーザー – Valkey または Redis OSSサーバーの認証メカニズム。詳細については、「」を参照してくださいAUTH。

> **Note**
>
> 4.0.2 の Redis OSSバージョンでは、転送中の暗号化が最初にサポートされ、6.0.4 が唯一のオプションAUTH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)
- 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. [Create] (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。

    このページは、クラスターがグローバルデータストアの一部であるかどうかを示すためにも更新されます。これには、以下が含まれます。

    - [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
- エンジンバージョン： グローバルデータストアを実行する Valkey または Redis OSS 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
- AUTH： 有効または無効

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が、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になっている Valkey または Redis の場合OSS、遅延はシャード間の最大遅延を秒単位で示します。

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信する別の AWS リージョンに読み取り専用クラスターを作成します。

グローバルデータストアに AWS リージョンを追加するには

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的に生成し、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、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
- 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)
   - 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
   - エンジンバージョンの変更 – Valkey 7.2 以降または Redis OSS エンジンバージョン 5.0.6 以降のみが使用できます。
   - [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。

   OSS クラスターモードが有効になっている Valkey または 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。この変更は、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 適用されます。そのクラスターのパラメータグループを変更するには、Valkey または Redis OSSコンソールまたは ModifyCacheClusterAPIオペレーションを使用します。詳細については、「 ElastiCache パラメータグループの変更」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、ResetCacheParameterGroupAPI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り、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 Valkey または Redis OSSでは、どちらが最終的にプライマリに解決されるかは保証されません。

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。

グローバルデータストアの使用 (CLI)

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、スクリプトを使用して自動化できます。は AWS CLI、アドホック (1 回限り) 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または Linux 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。

をダウンロード、インストール、設定するには CLI

1. AWS コマンドラインインターフェイスウェブページで 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 「 の設定CLI」の手順に従います。

グローバルデータストアでの AWS CLIの使用

グローバルデータストアを操作するには、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 は、作成時にグローバルデータストア ID にプレフィックス ElastiCache 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されて、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。

次の表に、各 AWS リージョンとそのグローバルデータストア ID プレフィックスを示します。

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	sgaui
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	quwsw

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 (ソウル) リージョン ap-northeast-2	lfqnh
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

リージョン名/リージョン	プレフィックス
中国 (寧夏) リージョン cn-northwest-1	ckbem
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、グローバルデータストアの名前を --global-replication-group-idパラメータに指定して、グローバルデータストア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値を渡すと、ElastiCache は、次のパラメータの 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
```

の Redis から OSS Valkey クロスエンジンへのアップグレード ElastiCache GlobalDataStore

コンソールAPIまたは を使用して、既存の Redis OSS グローバルレプリケーショングループを Valkey エンジンにアップグレードできますCLI。

既存の Redis OSS グローバルレプリケーショングループがある場合は、 で modify-global-replication-group新しいエンジンとエンジンバージョンを指定することで、Valkey にアップグレードできますAPI。

Linux、macOS、Unix の場合:

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id myGlobalReplGroup \
```

```
    --engine valkey \
    --apply-immediately \
    --engine-version 7.2
```

Windows の場合:

```
aws elasticache modify-global-replication-group ^
    --global-replication-group-id myGlobalReplGroup ^
    --engine valkey ^
    --apply-immediately ^
    --engine-version 7.2
```

アップグレードする既存の Redis OSS グローバルレプリケーショングループに適用されたカスタムキャッシュパラメータグループがある場合は、リクエストでカスタム Valkey キャッシュパラメータグループも渡す必要があります。入力 Valkey カスタムパラメータグループには、既存の Redis OSSカスタムパラメータグループと同じ Redis OSS静的パラメータ値が必要です。

Linux、macOS、Unix の場合:

```
aws elasticache modify-global-replication-group \
    --global-replication-group-id myGlobalReplGroup \
    --engine valkey \
    --engine-version 7.2 \
    --apply-immediately \
    --cache-parameter-group-name myParamGroup
```

Windows の場合:

```
aws elasticache modify-global-replication-group ^
    --global-replication-group-id myGlobalReplGroup ^
    --engine valkey ^
    --engine-version 7.2 ^
    --apply-immediately ^
    --cache-parameter-group-name myParamGroup
```

- 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-レプリケーショングループ

```
aws elasticache rebalance-shards-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name
```

Valkey または Redis ElastiCache で で使用できるすべてのコマンドを一覧表示するには、ヘルプを使用しますOSS。

```
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

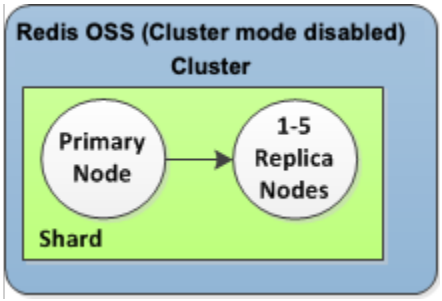
単一ノードの Amazon ElastiCache Valkey クラスターと Redis OSSクラスターは、データ保護サービスが制限されたインメモリエンティティです (AOF) 。クラスターが何らかの理由で停止すると、クラスターのすべてのデータが失われます。ただし、Valkey または Rediss OSS エンジンを実行している場合は、2～6 ノードをレプリカを持つクラスターにグループ化できます。1～5 つの読み取り専用ノードには、グループの単一の読み取り/書き込みプライマリノードのレプリケートデータが含まれます。このシナリオでは、1 個のノードが何らかの理由で停止した場合でも 1 個以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がありません。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。

次の図に示すように、レプリケーション構造は、Valkey または Redis OSSクラスター内に含まれるシャード (API/ ではノードグループと呼ばれますCLI) 内に含まれます。Valkey または Redis OSS (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。Valkey または Redis OSS (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をシャード間でパーティション分割して、最大 500 個のシャードを含めることができます。シャードの数が多くレプリカの数が少ないクラスターを作成できます。クラスターあたり最大 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。

ノードまたはシャードの制限は、Valkey および Redis OSS エンジンバージョン 5.0.6 以降では、クラスターごとに最大 500 まで引き上げることができます。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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることが含まれます。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。



Valkey または Redis OSS (クラスターモードが無効) クラスターには 1 つのシャードと 0 ~ 5 個のレプリカノードがあります

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、「Valkey または Redis の実行時の障害の軽減 OSS」を参照してください。

トピック

- Valkey と Redis OSSレプリケーションについて
- レプリケーション: Valkey および Redis OSS クラスターモード無効と有効
- Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS
- 同期とバックアップの実装方法
- Valkey または Redis OSS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
- Valkey または Redis OSS (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プライマリに昇格させる

Valkey と Redis OSSレプリケーションについて

Redis は、次の 2 つの方法でレプリケーションOSSを実装します。

- 各ノードのすべてのクラスターデータを含む単一のシャード — Valkey または Redis OSS (クラスターモードが無効)
- 最大 500 個のシャードに分割されたデータの場合 — Valkey または Redis OSS (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 OSS 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 に引き上げることができます。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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることがあります。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。

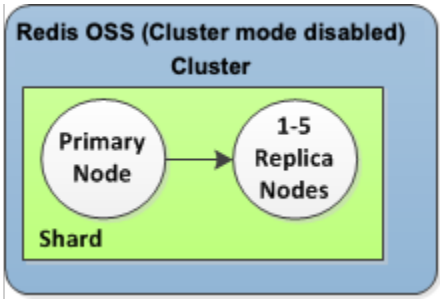
トピック

- Valkey または Redis OSS (クラスターモードが無効)
- Valkey または Redis OSS (クラスターモードが有効)

Valkey または Redis OSS (クラスターモードが無効)

Valkey または Redis OSS (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、その中にはノードのコレクションがあります。1 つのプライマリ読み取り/書き込みノードと最大 5 つのセ

カンダリ読み取り専用レプリカノードがあります。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。



単一のシャードノードとレプリカノードを持つ Valkey または Redis OSS (クラスターモードが無効) クラスター

レプリカノードで Valkey または Redis OSS (クラスターモードが無効) クラスターを使用すると、ソリューション ElastiCache を拡張して、読み取り集約型のアプリケーションを処理したり、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したりできます。

Valkey または Redis OSS (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、書き込みダウンタイムを減らすには、Valkey または Redis OSS (クラスターモードが無効) クラスターでマルチ AZ を自動フェイルオーバーで有効にします。詳細については、「Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS」を参照してください。

Valkey または Redis OSS (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は、プライマリロールとレプリカ交換ロールの 1 つを使用して変更できます。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、「Valkey または Redis OSS (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プライマリに昇格させる」を参照してください。

Valkey または Redis OSS (クラスターモードが有効)

Valkey または Redis OSS (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。

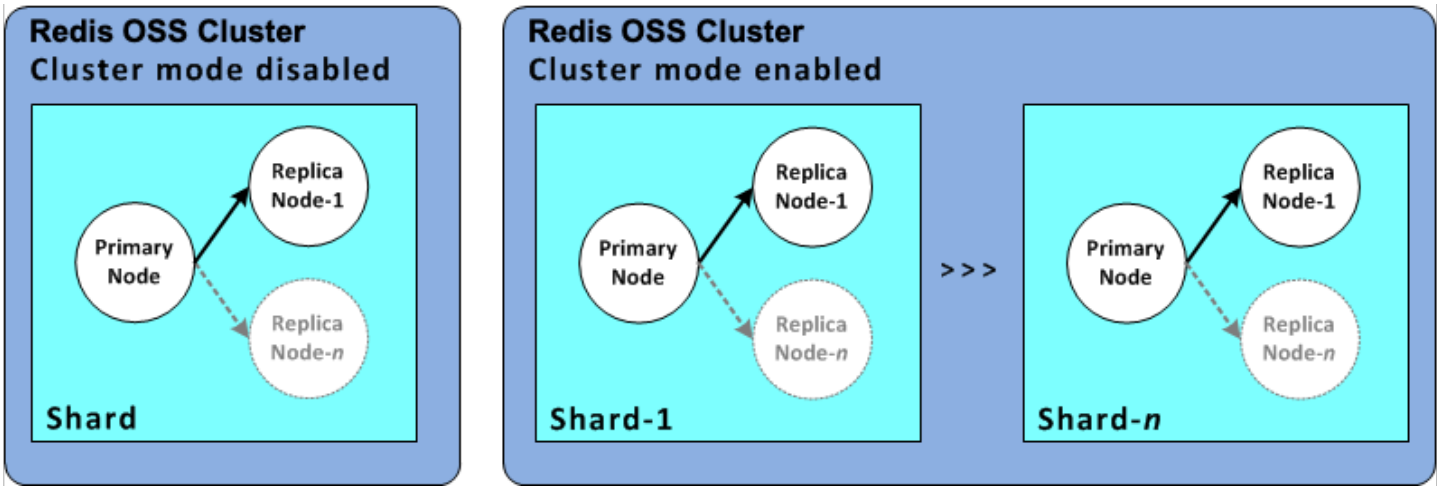
エンジンバージョンが Valkey 7.2 以降、または Redis 5.0.6 以降であれば、ノードまたはシャードの制限をクラスターごとに最大 OSS 500 に増やすことができます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることがあります。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、Valkey または Redis OSS (クラスターモードが有効) クラスターのシャード間でパーティション分割されます。

アプリケーションは、Valkey または Redis OSS (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に接続します。詳細については、「での接続エンドポイントの検索 ElastiCache」を参照してください。



複数のシャードとレプリカノードを持つ Valkey または Redis OSS (クラスターモードが有効) クラスター

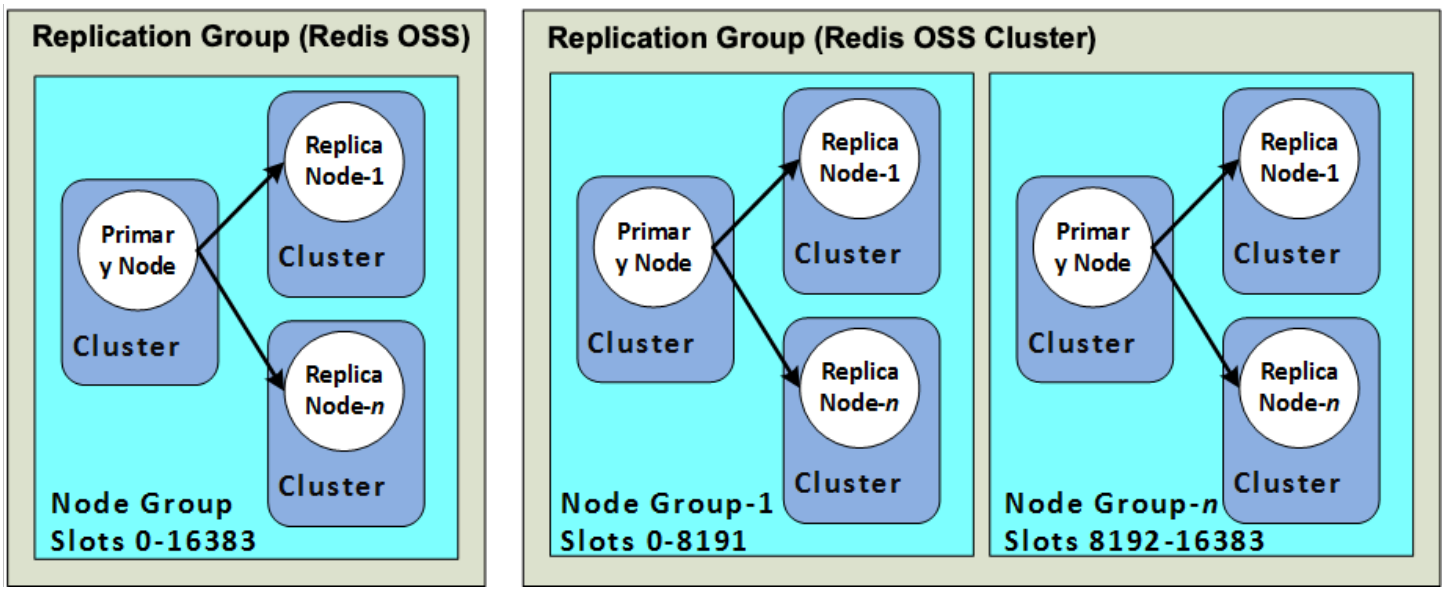
Valkey または Redis OSS (クラスターモードが有効) クラスター内のすべてのノードは、同じリージョンに存在する必要があります。耐障害性を向上させるために、そのリージョン内の複数のアベイラビリティーゾーンにプライマリとリードレプリカの両方をプロビジョニングできます。

現在、Valkey または Redis OSS (クラスターモードが有効) 機能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: Valkey および Redis OSS クラスターモード無効と有効

Valkey 7.2 および Redis OSSバージョン 3.2 以降では、2 つの異なるタイプのクラスター (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。Valkey または Redis OSS (クラスターモードが無効) クラスターには、常に最大 5 つのリードレプリカノードを持つ単一のシャード (API/CLI: ノードグループ) があります。Valkey または Redis OSS (クラスターモードが有効) クラスターには、それぞれ 1 ~ 5 個のリードレプリカノードを持つ最大 500 個のシャードがあります。



Valkey または Redis OSS (クラスターモードが無効) 、および Valkey または Redis OSS (クラスターモードが有効) クラスター

次の表は、Valkey または Redis OSS (クラスターモードが無効) クラスターと Valkey または Redis OSS (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たものです。

Valkey または Redis OSS (クラスターモードが無効) と Valkey または Redis OSS (クラスターモードが有効) クラスターの比較

機能	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (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、「のバージョン管理 ElastiCache」および「Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効) 」を参照してください。
データのパーティション化	不可	可能
シャード	1	1 ~ 500
リードレプリカ	0 ~ 5	シャードあたり 0 ~ 5。

機能	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (クラスターモードが有効)
	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。	**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。	可能 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。Valkey または Redis OSS (クラスターモードが無効) クラスターから単一の .rdb ファイルを使用します。	はい。Valkey または Redis OSS (クラスターモードが無効) または Valkey または Redis OSS (クラスターモードが有効) クラスターの .rdb ファイルを使用します。
サポート	すべての Valkey および Redis OSSバージョン	すべての Valkey OSS バージョン、および 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、「のバージョン管理 ElastiCache」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、「のバージョン管理 ElastiCache」を参照してください。

機能	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (クラスターモードが有効)
暗号化	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。
HIPAA 適格	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。
PCI DSS コンプライアンス	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。	バージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照してください) および 4.0.10 以降。
オンラインリシャーディング	該当なし	バージョン 3.2.10 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限終了スケジュール を参照してください) 以降。

どちらを使用すればよいですか?

Valkey または Redis OSS (クラスターモードが無効) または Valkey または Redis OSS (クラスターモードが有効) のいずれかを選択する場合は、次の要因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。Valkey または Redis OSS

(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、「」を参照してくださいValkey または Redis のレプリカノードのスケーリング OSS (クラスターモードが無効)。

Valkey または Redis OSS (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にデータをパーティション化できます。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の詳細については、「」を参照してくださいValkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)。

- ノードサイズとノード数 – Valkey または Redis OSS (クラスターモードが無効) クラスターにはシャードが 1 つだけあるため、ノードタイプは、クラスターのすべてのデータに加えて必要なオーバーヘッドに対応するのに十分な大きさである必要があります。一方、Valkey または Redis OSS (クラスターモードが有効) クラスターを使用する場合、複数のシャードにデータをパーティション化できるため、ノードタイプはさらに小さくなりますが、それ以上必要になります。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。

- Reads v. writes – クラスターのプライマリロードがデータの読み取り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、Valkey または Redis OSS (クラスターモードが無効) クラスターをスケーリングできます。ただし、リードレプリカの最大数は 5 です。クラスターの負荷が書き込みが多い場合は、複数のシャードを持つ Valkey または Redis OSS (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を利用できます。

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。

Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS

Valkey と Redis ElastiCache では、プライマリノードを置き換えるOSS必要があるインスタンスが多数あります。これには、特定のタイプの計画的なメンテナンスや、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の低いイベントが含まれます。

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。 ElastiCache はこれを透過的に処理するため、新しいプライマリノードを作成してプロビジョニングする必要はありません。このフェイルオーバーとレプリカの昇格により、昇格が完了したらすぐに新しいプライマリへの書き込みを再開できます。

ElastiCache は、昇格されたレプリカのドメインネームサービス (DNS) 名も伝達します。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 Valkey クラスターと Redis OSS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、計画されたノード置換が完了します。
- 5.0.6 以降のエンジンで実行されるマルチ AZ が有効になっている Valkey および Redis OSSクラスターモードが無効になっているクラスターの場合、クラスターが受信書き込みリクエストを処理する間、計画されたノード交換は完了します。
- 4.0.10 以前のエンジンで実行されるマルチ AZ が有効になっている Valkey および Redis OSSクラスターモードが無効になっているクラスターでは、DNS更新に関連する短時間の書き込み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 を有効にするには、 ElastiCache マネジメントコンソール、 AWS CLI、または を使用します ElastiCache API。

Valkey または Redis OSSクラスター (APIおよび 、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CLI、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を持つ Valkey クラスターと Redis OSSクラスターでのみサポートされています。

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
- マルチ AZ の制限事項

マルチ AZ の有効化

マルチ AZ は、 ElastiCache コンソール、または 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 AWS CLIまたは ElastiCache変更するときに有効にできますAPI。

マルチ AZ は、使用可能なリードレプリカが 1 つ以上ある Valkey または Redis OSS (クラスターモードが無効) クラスターでのみ有効にできます。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、「Valkey または Redis OSS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、「Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効) 」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)

ElastiCache コンソールを使用してマルチ AZ を有効にするには、新しい Valkey または Redis OSS クラスターを作成するとき、またはレプリケーションを使用して既存のクラスターを変更します。

マルチ AZ は、Valkey または Redis OSS (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っています。

> **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ー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、「Valkey (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 ElastiCache 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 AWS CLI 、 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。

> **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
```

このコマンドからの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
- modify-replication-group AWS CLI 「 コマンドリファレンス」の「」を参照してください。

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。

- CreateCacheCluster
- CreateReplicationGroup
- ModifyReplicationGroup

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、クラスター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、クラスター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
- クラスター全体が失敗した場合の障害シナリオ

プライマリノードのみが失敗した場合の障害シナリオ

プライマリノードのみが失敗した場合、レプリ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ー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DNSの名前を伝達します。
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ー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DNSの名前は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を加える必要はありません。
- [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。

- Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)
- Valkey または Redis OSS 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)
- Valkey または Redis OSS 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ーゾーンで、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 で昇格されたレプリカDNSの名前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ーゾーンで置き換えレプリカが作成されます。
4. すべての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DNSの名前は、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:

- Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)
- Valkey または Redis OSS 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)
- Valkey または Redis OSS 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:

- Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)
- Valkey または Redis OSS 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)
- Valkey または Redis OSS 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 を使用してテストできます ElastiCache API。

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 (および でノードグループと呼ばれます AWS CLI) の自動フェイルオーバーを ElastiCache APIテストできます。
- 異なるクラスター (APIおよび ではレプリケーショングループと呼ばれますCLI) のシャードに対してこのオペレーションを呼び出すと、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 Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループの異なるシャードで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 を使用してイベントを確認します ElastiCache API。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 ElastiCache APIリファレンスの
  - AWS CLI コマンドリファレンスの describe-events。
- これはAPI、 ElastiCache フェイルオーバー時の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は、この をブロック AWS することがありますAPI。

トピック

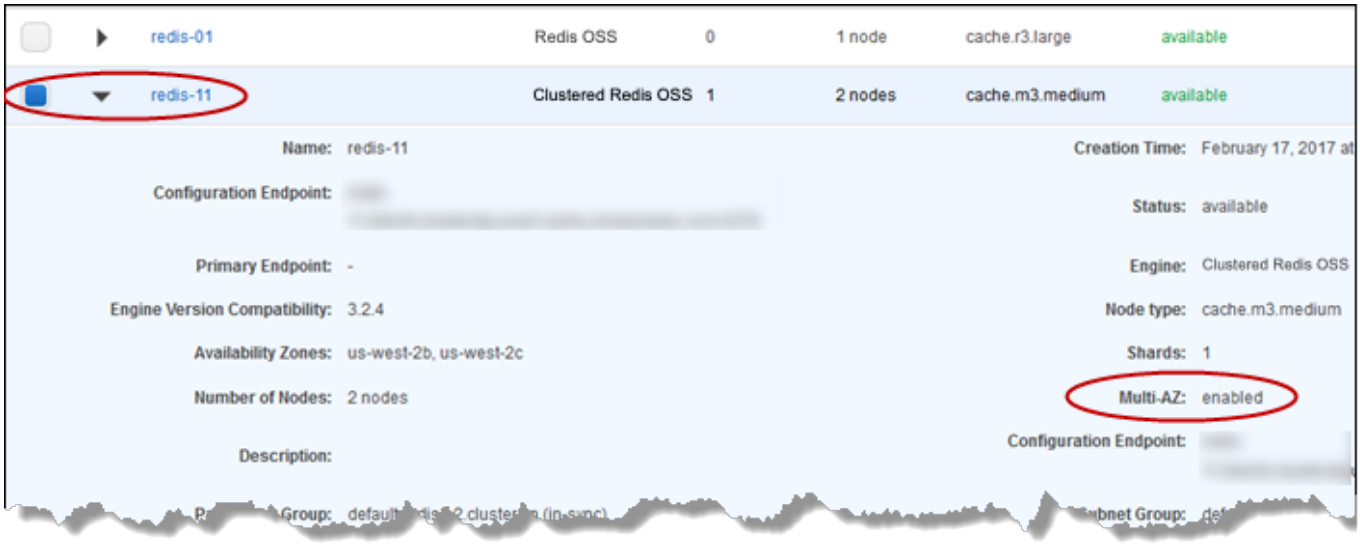
- を使用した自動フェイルオーバーのテスト AWS Management Console
- を使用した自動フェイルオーバーのテスト AWS CLI
- を使用した自動フェイルオーバーのテスト ElastiCache API

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、Valkey または Redis OSSを選択します。
3. クラスターのリストから、テストするクラスターの左側にあるボックスを選択します。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 ElastiCache AWS Management Console」を参照してください。



5. Valkey または Redis OSS (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。

   Valkey または Redis OSS (クラスターモードが有効) の場合は、以下を実行します。

a. クラスターの名前を選択します。

b. シャードページで、フェイルオーバーをテストするシャード (APIと でノードグループと呼ばれますCLI) に対して、シャードの名前を選択します。

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 対応クラスターの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 時間連続で最大 15 個のノードグループをテストできます。

次の例では AWS CLI 、 を使用して、Valkey または Redis OSS (クラスターモードが有効) クラスター `redis00-0003` のノードグループ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。

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 で有効なクラスターで自動フェイルオーバーを ElastiCache API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 時間連続で最大 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、 ElastiCache DescribeEventsAPI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

マルチ AZ の制限事項

マルチ AZ では、以下の制限に注意してください。

- マルチ AZ は Valkey および Redis OSSバージョン 2.8.6 以降でサポートされています。
- マルチ AZ は T1 ノードタイプではサポートされていません。
- Valkey レプリケーションと Redis OSSレプリケーションは非同期です。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場合、レプリケーションの遅延が最も少ないレプリカ ElastiCache を選択します。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。
- クラスターモードが無効になっている Valkey クラスターまたは Redis OSSクラスターでリードレプリカをプライマリに手動で昇格させる場合、マルチ AZ および自動フェイルオーバーが無効になっている場合にのみ行うことができます。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。これは、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オフにすることでコンソールから行うことができます。これは、 `ModifyReplicationGroup`オペレーションを呼び出す`false`ときに `AutomaticFailoverEnabled`プロパティを に設定 AWS CLI することでも実行できます。
  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 (Redis OSS) 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、使用可能である場合にのみ、AZ-a に新しいレプリカ ElastiCache を作成します (失敗したプライマリが配置されている) 。
- 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 Valkey と Redis にネイティブOSSであり、マルチ AZ に ElastiCache固有ではありません。この動作の詳細については、Valkey ウェブサイトの「レプリケーション」を参照してください。

> **Important**
>
> Valkey 7.2.6 以降または Redis OSSバージョン 2.8.22 以降では、外部レプリカを作成することはできません。
> 2.8.22 より前の Redis OSSバージョンでは、マルチ AZ が有効になっている ElastiCache クラスターに外部レプリカを接続しないことをお勧めします。このサポートされていない設定は、フェイルオーバーと復旧を適切に実行 ElastiCache できない問題を引き起こす可能性があります。外部レプリカを ElastiCacheクラスターに接続するには、接続を行う前にマルチ 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

Valkey と Redis でサポートされているすべてのバージョンでは、プライマリノードとレプリカノード間のバックアップと同期OSSがサポートされています。ただし、バックアップと同期の実装方法はバージョンによって異なります。

Redis OSSバージョン 2.8.22 以降

Redis OSSレプリケーションは、バージョン 2.8.22 以降では、2 つの方法から選択します。詳細については、「2.8.22 より前の Redis OSSバージョン」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。

2.8.22 より前の Redis OSSバージョン

2.8.22 より前のバージョンの Redis OSSバックアップと同期は 3 ステップのプロセスです。

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。これによりスナップショット point-in-timeが作成されます。
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、「Valkey または Redis OSS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を確認する」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。

Valkey または Redis OSS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を持つクラスターに関連付けられていない利用可能な Valkey または Redis OSS (クラスターモードが無効) クラスターが既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。現在、Valkey または Redis OSS (クラスターモードが有効) クラスターはゼロから作成する必要があります。

オプション 1: 既存の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

既存の単一ノードの Valkey または Redis OSS (クラスターモードが無効) クラスターを活用するには、このオプションを使用します。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。「既存の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**
>
> 既存のクラスターを使用して Valkey または Redis OSS (クラスターモードが有効) クラスターを作成することはできません。 ElastiCache コンソールを使用して Valkey または Redis OSS (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、「」を参照してくださいValkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール) 。

オプション 2: Valkey または Redis OSSレプリケーショングループをゼロから作成する

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できる Valkey または Redis OSS (クラスターモードが無効) クラスターをまだ持っていない場合、または Valkey または Redis OSS (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。「Valkey または Redis OSSレプリケーショングループをゼロから作成する」を参照してください。

既存の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、既存の単一ノードの Valkey または Redis OSSクラスターです。現在、Valkey または Redis OSS (クラスターモードが有効) は、使用可能な単一ノードクラスターを使用したレプリカを使用したクラスターの作成をサポートしていません。Valkey または Redis OSS (クラスターモードが有効) クラスターを作成する場合は、「」を参照してくださいValkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)。

以下の手順は、Valkey または Redis OSS (クラスターモードが無効) シングルノードクラスターがある場合にのみ使用できます。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 Valkey または Redis OSS (クラスターモードが無効) クラスターがない場合は、「」を参照してくださいValkey または Redis OSSレプリケーショングループをゼロから作成する。

既存の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「の使用 ElastiCache AWS Management Console」を参照してください。

使用可能な Valkey または Redis OSS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)

プライマリに使用可能な Valkey または Redis OSSCache クラスターを使用する場合、リードレプリカを使用して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 AWS CLI。

を使用する場合、 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`--primary-cluster-id`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 AWS CLI を作成しますCLI`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` と指定します。

Valkey または Redis OSS (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 Valkey または Redis OSS (クラスターモードが無効) クラスターのノードの名前。

次のコマンドは、使用可能な Valkey または Redis OSS (クラスターモードが無効) クラスターsample-repl-groupをレプリケーショングループのプライマリノードredis01として使用してレプリケーショングループを作成します。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

スタンドアロンの Valkey または Redis OSS (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)

を使用する場合 ElastiCache API、 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CLI`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 と指定します。

Valkey または Redis OSS (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。

ReplicationGroupDescription

レプリカを持つクラスターの説明。

NumCacheClusters

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

このクラスターのプライマリノードにする使用可能な Valkey または Redis OSS (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、使用可能な Valkey または Redis OSS (クラスターモードが無効) クラスターsample-repl-groupをレプリケーショングループのプライマリノードredis01として使用して、レプリカを持つクラスターを作成します。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、以下の ElastiCache APL のトピックを参照してください。

- CreateReplicationGroup
- ModifyReplicationGroup

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.

Valkey または Redis OSSレプリケーショングループをゼロから作成する

次に、既存の Valkey または Redis OSSクラスターをプライマリとして使用せずに Valkey または Redis OSSレプリケーショングループを作成する方法を示します。 ElastiCache コンソール、、または を使用して、Valkey または Redis OSS (クラスターモードが無効) 、または Valkey AWS CLIまたは Redis OSS (クラスターモードが有効) レプリケーショングループを ElastiCacheゼロから作成できますAPI。

続行する前に、Valkey または Redis OSS (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か、Valkey または Redis OSS (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: Valkey および Redis OSS クラスターモード無効と有効」を参照してください。

トピック

- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する
- Valkey または Redis OSS (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ゼロから作成する

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する

コンソール、、または を使用して ElastiCache、最初から Valkey AWS CLIまたは Redis OSS (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できます ElastiCache API。Valkey または Redis OSS (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、1 つのプライマリクラスター、最大 5 つのリードレプリカがあります。Valkey または Redis OSS (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。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成するには、次のいずれかの方法を実行します。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する (AWS CLI)

次の手順では、 を使用して Valkey または Redis OSS (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 AWS CLI。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する場合は、レプリケーショングループとそのすべてのノードを `create-replication-group` コマンドを 1 回呼び出すだけで作成します AWS CLI 。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。

Valkey または Redis OSS (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、「のデータ階層化 ElastiCache」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 OSS3.2.4 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 から派生したパラメータグループを指定`default.redis3.2`して、Valkey または Redis OSS (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。詳細については、「Valkey パラメータと Redis OSS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 OSSバージョン 3.2.6 または 4.0.10 を実行している必要があります。
- レプリケーショングループは Amazon で作成する必要がありますVPC。
- パラメータ --cache-subnet-group も含める必要があります。
- また、このレプリケーショングループで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) の顧客指定の文字列値--auth-tokenに パラメータを含める必要があります。

次のオペレーションでは、プライマリレプリカと 2 つのレプリカの 3 つのノードsample-repl-groupを持つ Valkey または Redis OSS (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。

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。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する (ElastiCache API)

次の手順では、 を使用して Valkey または Redis OSS (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 ElastiCache API。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成すると、レプリケーショングループとそのすべてのノードが CreateReplicationGroupオペレーションを ElastiCache API 1 回呼び出すだけで作成されます。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。

Valkey または Redis OSS (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。

ReplicationGroupDescription

レプリケーショングループの説明。

NumCacheClusters

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。

CacheNodeType

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、「のデータ階層化 ElastiCache」を参照してください。

CacheParameterGroup

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 OSS3.2.4 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 から派生したパラメータグループを指定`default.redis3.2`して、Valkey または Redis OSS (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。詳細については、「Valkey パラメータと Redis OSS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 OSSバージョン 3.2.6 または 4.0.10 を実行している必要があります。
- レプリケーショングループは Amazon で作成する必要がありますVPC。
- パラメータ CacheSubnetGroup も含める必要があります。
- また、このレプリケーショングループで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) の顧客指定の文字列値AuthTokenに パラメータを含める必要があります。

次のオペレーションでは、プライマリレプリカと 2 つのレプリカの 3 つのノードmyReplGroupを持つ Valkey または Redis OSS (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。

```
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.

Valkey または Redis OSS (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ゼロから作成する

ElastiCache コンソール、、または を使用して、Valkey または Redis OSS (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) AWS CLIを作成できます ElastiCache API。Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループには、1～500 個のシャード (API/CLI: ノードグループ) 、各シャードのプライマリノード、各シャードのリードレプリカが最大 5 個あります。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。

Valkey または Redis OSS エンジンのバージョンが 5.0.6 以上であれば、ノードまたはシャードの制限をクラスターごとに最大 500 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。一般的な落とし穴には、サブネットグループのサブネットCIDRの範囲が小さすぎるか、サブネットが共有され、他のクラスターによって頻繁に使用されることがあります。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。

Valkey または Redis でのクラスターの作成 OSS (クラスターモードが有効)

- Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)
- Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成する (AWS CLI)
- Valkey または Redis OSS (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ゼロから作成する (ElastiCache API)

Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)

Valkey または Redis OSS (クラスターモードが有効) クラスターを作成するには、「」を参照してくださいValkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)。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。

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成する (AWS CLI)

次の手順では、 を使用して Valkey または Redis OSS (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 AWS CLI。

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成するときは、レプリケーショングループとそのすべてのノードを `create-replication-group` コマンドを 1 回呼び出すだけで作成します AWS CLI 。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。

Valkey または Redis OSS (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、「のデータ階層化 ElastiCache」を参照してください。

--cache-parameter-group

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループdefault.redis6.x.cluster.onを作成するには、 default.redis6.x.cluster.on から派生したパラメータグループを指定します。詳細については、「Redis 6.x OSS 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 OSSバージョン 3.2.6 または 4.0.10 を実行している必要があります。
- レプリケーショングループは Amazon で作成する必要がありますVPC。

- パラメータ --cache-subnet-group も含める必要があります。
- また、このレプリケーショングループで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) の顧客指定の文字列値--auth-tokenに パラメータを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/シャード (--) sample-repl-groupを持つ Valkey または Redis OSS (クラスターモードが有効num-node-groups) レプリケーショングループを作成します。それぞれに 3 つのノード、1 つのプライマリレプリカと 2 つのリードレプリカ (--) がありますreplicas-per-node-group。

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": {}
    }
}
```

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成すると、次の 2 つのノードグループ (コンソール: シャード) を設定する例に示すように、 `--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ーゾーン。このパラメータを省略すると、 はプライマリノードのアベイラビリティーゾーン ElastiCache を選択します。

  例: us-west-2a。

- `ReplicaAvailabilityZones` – リードレプリカがあるアベイラビリティー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ーゾーンの数は、`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ーゾーン 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に ElastiCache 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/シャード OSS () new-groupを持つ Valkey 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作成します--num-node-groups。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.

Valkey または Redis OSS (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ゼロから作成する (ElastiCache API)

次の手順では、 を使用して Valkey または Redis OSS (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 ElastiCache API。

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成するときは、レプリケーショングループとそのすべてのノードを CreateReplicationGroupオペレーションを 1 回呼び出すだけで作成します ElastiCache API。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。

Valkey または Redis OSS (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。

ReplicationGroupDescription

レプリケーショングループの説明。

NumNodeGroups

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。

NodeGroupConfiguration

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ーゾーン。このパラメータを省略すると、 はプライマリノードのアベイラビリティーゾーン ElastiCache を選択します。

  例: us-west-2a。
- ReplicaAvailabilityZones – リードレプリカがあるアベイラビリティーゾーンのリスト。このリストのアベイラビリティーゾーンの数は、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ーゾーン 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に ElastiCache 割り当てます。

  例: "0-4999"

CacheNodeType

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、「のデータ階層化 ElastiCache」を参照してください。

CacheParameterGroup

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループdefault.redis6.x.cluster.onを作成するには、 default.redis6.x.cluster.on から派生したパラメータグループを指定します。詳細については、「Redis 6.x OSS 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 OSSバージョン 3.2.6 または 4.0.10 を実行している必要があります。
- レプリケーショングループは Amazon で作成する必要がありますVPC。
- パラメータ CacheSubnetGroup も含める必要があります。
- また、このレプリケーショングループで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) の顧客指定の文字列値AuthTokenに パラメータを含める必要があります。

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=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParemeterGroup=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>
```

使用する追加情報とパラメータについては、「」トピックを参照してください。 ElastiCache APICreateReplicationGroup.

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 用 ElastiCache、または を使用できます ElastiCache API。コンソールプロセスは、Valkey または Redis OSS (クラスターモードが無効) と Valkey または Redis OSS (クラスターモードが有効) で異なります。

レプリケーショングループの詳細の表示

- レプリカを使用した Valkey または Redis OSS (クラスターモードが無効) の表示
  - Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (コンソール)
  - Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (AWS CLI)
  - Valkey または Redis の表示 OSS (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)
- レプリケーショングループの表示: Valkey または Redis OSS (クラスターモードが有効)
  - Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (コンソール)
  - Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (AWS CLI)

  - Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (ElastiCache API)
- レプリケーショングループの詳細の表示 (AWS CLI)
- レプリケーショングループの詳細の表示 (ElastiCache API)

レプリカを使用した Valkey または Redis OSS (クラスターモードが無効) の表示

ElastiCache コンソール、 AWS CLI for 、または を使用して、レプリカ OSS (API/CLI: レプリケーショングループ) を持つ Valkey ElastiCacheまたは 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 ElastiCache API。

Valkey または Redis OSS (クラスターモードが無効) クラスターの詳細の表示

- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (コンソール)
- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (AWS CLI)
- Valkey または Redis の表示 OSS (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)

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (コンソール)

ElastiCache コンソールを使用してレプリカを持つ Valkey または Redis OSS (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Valkey または Redis OSS (クラスターモードが無効) の詳細の表示 (コンソール)。

Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの表示 (AWS CLI)

Valkey または Redis OSS (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)。

Valkey または Redis の表示 OSS (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)

Valkey ElastiCache APIまたは Redis OSS (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表示: Valkey または Redis OSS (クラスターモードが有効)

Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (コンソール)

ElastiCache コンソールを使用して Valkey または Redis OSS (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Valkey または Redis OSS (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) 。

Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (AWS CLI)

Valkey ElastiCache CLIまたは Redis OSS (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)。

Valkey または Redis OSS (クラスターモードが有効) クラスターの表示 (ElastiCache API)

Valkey ElastiCache APIまたは Redis OSS (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)

コマンドを使用して AWS CLI `describe-replication-groups`、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、次のようになります (JSON 形式) 。

```
{
   "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、「 トピック AWS CLI ElastiCache 」の「」を参照してください。describe-replication-groups.

レプリケーショングループの詳細の表示 (ElastiCache API)

オペレーションを使用して AWS CLI DescribeReplicationGroups、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、「 リファレンストピック ElastiCache API」を参照してください。DescribeReplicationGroups.

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、そのノードの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の任意のノードに接続できます。Valkey または Redis OSS (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、Valkey または Redis OSS (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応じて、さまざまなエンドポイントに関心があります。

Valkey または Redis OSS (クラスターモードが無効)

レプリカを持つ Valkey または Redis OSS (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 、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、クラスター内のプライマリノードに常に解決されるDNS名前です。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着信接続を ElastiCache クラスター内のすべてのリードレプリカに均等に分割します。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 (Redis OSS) 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。

Valkey または Redis OSS (クラスターモードが有効)

レプリカを持つ Valkey または Redis OSS (クラスターモードが有効) クラスターは、複数のシャード (API/CLI: ノードグループ) を持つため、複数のプライマリノードも持つため、Valkey または Redis OSS (クラスターモードが無効) クラスターとは異なるエンドポイント構造を持ちます。Valkey また

は Redis OSS (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「認識」す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。アプリケーションが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、Valkey と Redis はOSS、シーンの背後にあるキーが属するシャードとそのシャード内のどのエンドポイントを使用するかを決定します。これはすべてアプリケーションに対して透過的です。

ElastiCache コンソール、、または を使用して AWS CLI、クラスターのエンドポイントを見つけることができます 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

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。

- Valkey または Redis OSS (クラスターモードが無効) クラスターのエンドポイントの検索 (コンソール)
- Valkey または Redis OSS (クラスターモードが有効) クラスターのエンドポイントの検出 (コンソール)
- Valkey または Redis OSS 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)
- Valkey または Redis OSS 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 ModifyReplicationGroup (CLI:) を使用して エンジンバージョンを変更するなど、Valkey または Redis OSS (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 (CLI:) を使用して、Valkey または Redis OSS (クラスターモードが有効) クラスター内のシャード ModifyReplicationGroupShardConfiguration (ノードグループ) の数を変更できますmodify-replication-group-shard-configuration。詳細については、「Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効) 」を参照してください。

  Valkey または Redis OSS (クラスターモードが有効) クラスターへのその他の変更では、変更を組み込んだ新しいクラスターでクラスターを作成する必要があります。
- Valkey または Redis OSS (クラスターモードが無効) および Valkey または Redis OSS (クラスターモードが有効) 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、「のバージョン管理 ElastiCache」を参照してください。
- 以下の例に示すように、 コンソールまたは modify-replication-group CLI コマンドを使用して、クラスターモードを無効にするクラスターモードOSSを使用する Valkey ModifyReplicationGroupAPIまたは Redis クラスター ElastiCache で既存の をアップグレードできます。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。

ElastiCache コンソール、、または を使用して、Valkey または Redis OSS (クラスターモードが無効) クラスターの設定を変更できます AWS CLI ElastiCache API。現在、 は Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループで、限定された数の変更 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 Valkey または Redis OSS (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
- の使用 ElastiCache API

の使用 AWS Management Console

Valkey または Redis OSS (クラスターモードが無効) クラスターを変更するには、「」を参照してください ElastiCache クラスターの変更。

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 Valkey または Redis OSSレプリケーショングループでマルチ AZ を有効にします。

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、Valkey または Redis OSSクライアントは、クラスターモードが有効とクラスターモードが無効の両方を使用して接続できます。すべての Valkey または Redis OSS クライアントをクラスターモードが有効になるように移行したら、クラスターモード設定を完了し、クラスターモードを有効 に設定することができます。

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`「」を参照してください。modify-replication-group または ElastiCache (Redis OSS) ユーザーガイド のクラスターモードの変更。

の使用 ElastiCache API

次の ElastiCache APIオペレーションでは、既存の Valkey または Redis OSS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
    &Mutli-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>
```

ModifyReplicationGroup オペレーションの詳細については、 ElastiCache API「」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/ ではレプリケーショングループと呼ばれますCLI) の 1 つが不要になった場合は、削除できます。レプリケーショングループを削除すると、 はそのグループ内のすべてのノード ElastiCache を削除します。

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 (Redis OSS) クラスターを削除すると、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。
> - `CreateSnapshot` 最終スナップショットを作成するには、 アクセス許可が必要です。このアクセス許可がない場合、API呼び出しは`Access Denied`例外で失敗します。

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、「でのクラスターの削除 ElastiCache」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)

コマンドを使用する delete-replication-group レプリケーショングループを削除します。

```
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)

電話 DeleteReplicationGroup `ReplicationGroup` パラメータを使用します。

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

、、または を使用して AWS Management Console、Valkey または Redis レOSSプリケーショングループのリードレプリカの数を動的 AWS CLIに増減できます ElastiCache API。レプリケーショングループが Valkey または Redis OSS (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は、レプリカの数を増減するシャード (ノードグループ) を選択できます。

レプリケーショングループ内のレプリカの数を動的に変更するには、次の表から状況に合ったオペレーションを選択します。

目的	Valkey または Redis の場合 OSS (クラスターモードが有効)	Valkey または Redis の場合 OSS (クラスターモードが無効)
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 Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効)
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 Valkey または Redis のリードレプリカの削除 OSS (クラスターモードが無効)

シャードのレプリカの数を増やす

Valkey または Redis OSS (クラスターモードが有効) シャード、Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループのレプリカの数を最大 5 つ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 を使用します ElastiCache API。

トピック

- の使用 AWS Management Console
- の使用 AWS CLI
- の使用 ElastiCache API

の使用 AWS Management Console

次の手順では、 コンソールを使用して、Valkey または Redis OSS (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の数を増やします。

シャード内のレプリカの数を増やすには

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、Valkey または Redis OSS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 つにすることをお勧めします。
    - アベイラビリティーゾーン では、新しいレプリカごとにアベイラビリティーゾーン ElastiCache を選択する設定なしを選択するか、アベイラビリティーゾーンを指定して新しいレプリカごとにアベイラビリティーゾーンを選択します。

      [Specify Availability Zones (アベイラビリティーゾーンを指定する)] 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ー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。

の使用 AWS CLI

Valkey シャードまたは Redis OSS シャード内のレプリカの数を増やすには、次のパラメータで `increase-replica-count` コマンドを使用します。

- `--replication-group-id` – 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。
- `--apply-immediately` または `--no-apply-immediately` – 必須。レプリカの数をすぐに増やすか (`--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。このパラメータは、ノードグループが 1 つしかない場合、またはすべてのノードグループに同じ数のレプリカを持たせたい場合、Valkey または Redis OSS (クラスターモードOSSが無効) レプリケーショングループに使用します。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ーゾーンを個別に設定できます。このパラメータは、各ノードグループを個別に設定する Valkey または Redis OSS (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。Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの場合、シャード ID は常に です`0001`。Valkey または Redis OSS (クラスターモードが有効) ノードグループの (シャード) ID を検索するには、「」を参照してくださいシャードの ID を見つける。
  - `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ーゾーンを指定する `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 のメンバー`--replica-configuration`を省略すると、 ElastiCache (Redis OSS) は新しいレプリカごとにアベイラビリティー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を持つ Valkey または Redis OSS (クラスターモードが無効) グループであるか、複数のノードグループを持つ Valkey または Redis OSS (クラスターモードが有効) グループであるかにかかわらず適用されます。

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存在する場合、これは Valkey または Redis OSS (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ーゾーンの数は、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 \
        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,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,us-east-1c \
    --apply-immediately
```

を使用してレプリカの数を増やす方法の詳細についてはCLI、「Amazon コマンドラインリファレンスincrease-replica-count」の「」を参照してください。 ElastiCache

の使用 ElastiCache API

Valkey シャードまたは Redis OSS シャード内のレプリカの数を増やすには、次のパラメータで 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。レプリカの数をすぐに増やすか (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。このパラメータは、ノードグループが 1 つしかない Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループ、またはすべてのノードグループに同じ数のレプリカを持たせる Valkey または Redis OSS (クラスターモードが有効) グループに使用します。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ーゾーンを個別に設定できます。このパラメータは、各ノードグループを個別に設定する Valkey または Redis OSS (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。Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの場合、ノードグループ (シャード) ID は常に です0001。Valkey または Redis OSS (クラスターモードが有効) ノードグループの (シャード) ID を検索するには、「」を参照してくださいシャードの ID を見つける。
- 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ーゾーンを指定する 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 のメンバーReplicaConfigurationを省略すると、 ElastiCache (Redis OSS) は新しいレプリカごとにアベイラビリティー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を持つ Valkey または Redis OSS (クラスターモードが無効) グループであるか、複数のノードグループを持つ Valkey または Redis OSS (クラスターモードが有効) グループであるかにかかわらず適用されます。

```
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存在する場合、これは Valkey または Redis OSS (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ーゾーンの数は、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。

```
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>
```

を使用してレプリカの数を増やす方法の詳細についてはAPI、「Amazon リファレンスIncreaseReplicaCount」の「」を参照してください。 ElastiCache API

シャードのレプリカの数を減らす

Valkey または Redis のシャード OSS (クラスターモードが有効) または Valkey または Redis のレプリケーショングループ OSS (クラスターモードが無効) のレプリカの数を減らすことができます。

- Valkey または Redis OSS (クラスターモードが無効) では、マルチ AZ が有効になっている場合はレプリカの数を 1 に、有効になっていない場合はゼロに減らすことができます。
- Valkey または Redis OSS (クラスターモードが有効) では、レプリカの数をゼロに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内のレプリカの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
- の使用 ElastiCache API

の使用 AWS Management Console

次の手順では、 コンソールを使用して、Valkey または Redis OSS (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の数を減らします。

Valkey または Redis OSS シャード内のレプリカの数を減らすには

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、Valkey または Redis OSSを選択し、レプリカを削除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の数を入力します。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、 ElastiCache (Redis OSS) は自動的に削除するレプリカノードを選択します。その間、 ElastiCache (Redis OSS) 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ようとし、その後、プライマリでレプリケーションの遅延を最小限に抑えてレプリカを保持します。
> - 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

Valkey シャードまたは Redis OSS シャード内のレプリカの数を減らすには、次のパラメータで `decrease-replica-count` コマンドを使用します。

- `--replication-group-id` – 必須。レプリカの数を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。レプリカの数をすぐに減らすか (`--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を減らす」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IDsを指定するノード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ーゾーンを個別に設定できます。このパラメータは、各ノードグループを個別に設定する Valkey または Redis OSS (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。Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの場合、シャード ID は常に です0001。Valkey または Redis OSS (クラスターモードが有効) ノードグループの (シャード) ID を検索するには、「」を参照してくださいシャードの ID を見つける。
- 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を減らす」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ーゾーンを指定する 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 のメンバー--replica-configurationを省略すると、 ElastiCache (Redis OSS) は新しいレプリカごとにアベイラビリティー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の数を 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を持つ Valkey または Redis OSS (クラスターモードが無効) グループであるか、複数のノードグループを持つ Valkey または Redis OSS (クラスターモードが有効) グループであるかにかかわらず適用されます。

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を減らします。

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存在する場合、これは Valkey または Redis OSS (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ーゾーンの数は、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 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
```

を使用してレプリカの数を減らす方法の詳細についてはCLI、「Amazon コマンドラインリファレンスdecrease-replica-count」の「」を参照してください。 ElastiCache

の使用 ElastiCache API

Valkey シャードまたは Redis OSS シャード内のレプリカの数を減らすには、次のパラメータで DecreaseReplicaCountアクションを使用します。

- ReplicationGroupId – 必須。レプリカの数を減らすレプリケーショングループを指定します。
- ApplyImmediately – 必須。レプリカの数をすぐに減らすか (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を減らす」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IDsを指定するノード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ー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。このパラメータは、各ノードグループを個別に設定する Valkey または Redis OSS (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。Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループの場合、ノードグループ ID は常に です0001。Valkey

または Redis OSS (クラスターモードが有効) ノードグループの (シャード) ID を検索するには、「」を参照してくださいシャードの ID を見つける。

- 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ーゾーンを指定する 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 のメンバーReplicaConfigurationを省略すると、 ElastiCache (Redis OSS) は新しいレプリカごとにアベイラビリティー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の数を 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を持つ Valkey または Redis OSS (クラスターモードが無効) グループであるか、複数のノードグループを持つ Valkey または Redis OSS (クラスターモードが有効) グループであるかにかかわらず適用されます。

```
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の数を減らします。

```
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存在する場合、これは Valkey または Redis OSS (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ーゾーンの数は、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。

```
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>
```

を使用してレプリカの数を減らす方法の詳細についてはAPI、「Amazon リファレンスDecreaseReplicaCount」の「」を参照してください。 ElastiCache API

Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効)

次のトピックの情報は、Valkey または Redis OSS (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、Valkey または Redis OSS (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。

Valkey または Redis OSS (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、最大 5 つのリードレプリカを含めることができます。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と、オペレーションが失敗します。

Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループにレプリカを追加する方法については、以下を参照してください。

- Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)
- シャードのレプリカの数を増やす

ElastiCache コンソール、、または を使用して、Valkey または Redis OSS (クラスターモードが無効) AWS CLIクラスターにリードレプリカを追加できます ElastiCache API。

関連トピック

- ElastiCache 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)
- を使用して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 API

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)

Valkey または Redis OSS (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、 コマンドと パラメータを使用して AWS CLI `create-cache-cluster`、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`--replication-group-id`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
```

を使用してリードレプリカを追加する方法の詳細についてはCLI、「」を参照してください。create-cache-cluster「Amazon ElastiCache コマンドラインリファレンス」の「」を参照してください。

を使用して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 API

リードレプリカを Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループに追加するには、 ElastiCache `CreateCacheCluster`オペレーションと パラメータを使用して、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`ReplicationGroupId`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>
```

を使用してリードレプリカを追加する方法の詳細についてはAPI、「」を参照してくださ
い。CreateCacheCluster「Amazon ElastiCache API リファレンス」の「」を参照してください

Valkey または Redis のリードレプリカの削除 OSS (クラスターモードが無効)

次のトピックの情報は、Valkey または Redis OSS (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。

Valkey または Redis OSSレプリケーショングループのリードトラフィックが変更されると、リードレプリカを追加または削除できます。 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。

  1. 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、「Valkey または Redis OSS (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プライマリに昇格させる」を参照してください。
  2. 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。

  1. 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。
  2. リードレプリカを削除します。

ElastiCache コンソール、 AWS CLI for 、 ElastiCacheまたは を使用して、Valkey または Redis OSS (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できます ElastiCache API。

Valkey または Redis OSSレプリケーショングループからクラスターを削除する手順については、以下を参照してください。

- の使用 AWS Management Console
- AWS CLI を使用して ElastiCache クラスターを削除する
- の使用 ElastiCache API
- Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)
- シャードのレプリカの数を減らす

Valkey または Redis OSS (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プライマリに昇格させる

次のトピックの情報は、Valkey または Redis OSS (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。

、、または を使用して、Valkey または Redis OSS (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 AWS Management Console AWS CLI ElastiCache API。 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。Valkey または Redis OSS (クラスターモードが無効) レプリカをマルチ AZ 対応レプリケーショングループのプライマリに昇格させるには、以下を実行します。

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 OSS2.6.13 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. 昇格するレプリカがマルチ AZ が有効になっている Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループのメンバーである場合は、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. Valkey または Redis OSSを選択し、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、クラスターの次回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ーでマルチ 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 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、リードレプリカをレプリケーショングループのmy-replica-1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、「」を参照してください。modify-replication-group「Amazon ElastiCache コマンドラインリファレンス」の「」を参照してください。

の使用 ElastiCache API

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、リードレプリカをレプリケーショングループのmyReplica-1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、「」を参照してください。ModifyReplicationGroup「Amazon ElastiCache API リファレンス」の「」を参照してください

ElastiCache クラスターメンテナンスの管理

すべてのクラスターには、週ごとのメンテナンス時間があります。その時間内にシステムの変更が適用されます。Valkey と Redis ではOSS、レプリケーショングループには同じ週次メンテナンスウィンドウがあります。クラスターまたはレプリケーショングループを作成または変更するときに優先メンテナンスウィンドウを指定しない場合、 はランダムに選択された曜日にリージョンのメンテナンスウィンドウ内に 60 分のメンテナンスウィンドウを ElastiCache 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	欧州 (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

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-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 ~ 6:00 UTC
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 (UAE) 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**
>
> 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すぐに適用する場合は、今すぐ適用ボックス AWS Management Console 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。を使用するには API、modify-replication-group「」または「」を参照してくださいmodify-cache-cluster。

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 —FAQ メンテナンスとノード交換
- ノードの交換 (Memcached)— Memcached のノード置換の管理
- ElastiCache クラスターの変更—クラスターのメンテナンスウィンドウの変更
- ノードの交換 (Valkey と Redis OSS) —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 ElastiCache 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ランタイム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。Memcached パラメータのテーブルについては、「」を参照してくださいMemcached 固有のパラメータ。Valkey パラメータと Redis OSSパラメータのテーブルについては、「」を参照してくださいValkey パラメータと Redis OSSパラメータ。

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。ノードタイプ別のこれらの Memcached パラメータ値の表については、「」を参照してくださいMemcached ノードタイプ固有のパラメータ。ノードタイプ別のこれらの Valkey パラメータ値と Redis OSSパラメータ値の表については、「」を参照してくださいRedis OSSノードタイプ固有のパラメータ。

> **Note**
>
> Memcached 固有のパラメータの一覧については、「Memcached 固有のパラメータ」を参照してください。

トピック

- でのパラメータ管理 ElastiCache
- でパラメータグループ階層をキャッシュする ElastiCache

- ElastiCache パラメータグループの作成
- 名前による ElastiCache パラメータグループの一覧表示
- ElastiCache パラメータグループの値を一覧表示する
- ElastiCache パラメータグループの変更
- ElastiCache パラメータグループの削除
- エンジン固有のパラメータ

でのパラメータ管理 ElastiCache

ElastiCache パラメータは、パラメータ管理を容易にするために、名前付きパラメータグループにグループ化されます。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。
- Memcached の場合、パラメータグループファミリーと割り当てるクラスターは互換性がある必要があります。たとえば、クラスターが Memcached バージョン 1.4.8 を実行している場合は、Memcached 1.4 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。

  Redis の場合、パラメータグループファミリーと割り当てるクラスターは互換性がある必要があります。例えば、クラスターが Redis OSSバージョン 3.2.10 を実行している場合は、Redis OSS3.2 ファミリーのパラメータグループ、デフォルトまたはカスタム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。
- Memcached の場合、クラスターのパラメータを変更すると、変更がクラスターに直ちに適用されます。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、Memcached 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。クラスターのノードの再起動については、「クラスターの再起動」を参照してください。
- Redis の場合、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ます。ただし、クラスターノードを再起動した後は、以下の例外を除きます。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、Valkey パラメータと Redis OSS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。

Valkey または Redis OSSノードの再起動の詳細については、「」を参照してください[ノードの再起動](#)。

> **Valkey または Redis OSS (クラスターモードが有効) パラメータの変更**
>
> Valkey または Redis OSS (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、次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. 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。

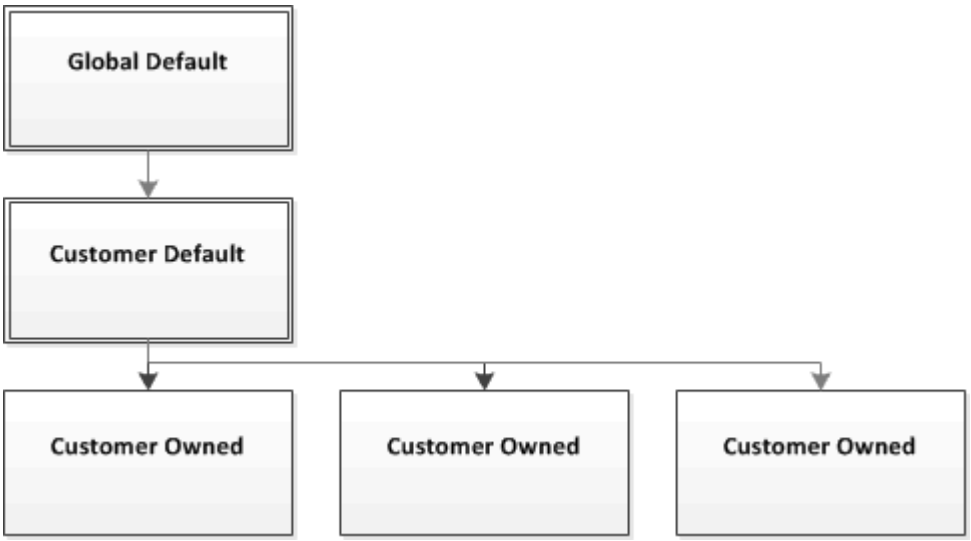
- パラメータグループを Valkey および Redis OSS グローバルデータストアに関連付けることができます。グローバルデータストアは、 AWS リージョンにまたがる 1 つ以上のクラスターのコレクションです。この場合、パラメータグループは、Global Datastore を構成するすべてのクラスターで共有されます。プライマリクラスターのパラメータグループに対する変更は、Global Datastore 内の残りのすべてのクラスターにレプリケートされます。詳細については、「[グローバルデータストアを使用した AWS リージョン間のレプリケーション](#)」を参照してください。

  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、はい/いいえ グローバル属性
  - [CacheParameterGroup](#) API オペレーションのはい/いいえ`IsGlobal`プロパティ

でパラメータグループ階層をキャッシュする ElastiCache

Amazon ElastiCache には、次に示すように 3 つのキャッシュパラメータグループがあります。



Amazon ElastiCache パラメータグループ階層

グローバルデフォルト

リージョン内のすべての Amazon ElastiCache 顧客の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:

- 用に予約 ElastiCache されており、お客様が利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:

- によって作成および所有されます ElastiCache。
- 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。Memcached 値の詳細については、「」を参照してくださいMemcached 固有のパラメータ。Valkey と Redis OSSの値の詳細については、「」を参照してくださいValkey パラメータと Redis OSSパラメータ。

ElastiCache パラメータグループの作成

デフォルト値から変更するパラメータの値が 1 つ以上ある場合、新しいパラメータグループを作成する必要があります。パラメータグループは、ElastiCache コンソール、、 AWS CLIまたは を使用して作成できます ElastiCache API。

ElastiCache パラメータグループの作成 (コンソール)

次の手順は、 コンソールを使用して ElastiCacheパラメータグループを作成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. 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。

   memcached1.4 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と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で選択します。したがって、わかりやすくパラメータグループのファミリーを特定するのに役立つ名前をお勧めします。

   パラメータグループの命名に関する制約は次のとおりです。

   - ASCII 文字で始まる必要があります。
   - 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、作成 を選択します。

   パラメータグループを作成しないでプロセスを終了するには、Cancel を選択します。
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、「ElastiCache パラメータグループの変更」を参照してください。

ElastiCache パラメータグループの作成 (AWS CLI)

を使用してパラメータグループを作成するには AWS CLI、これらのパラメータ`create-cache-parameter-group`で 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。

  - ASCII 文字で始まる必要があります。
  - 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、memcachedmyMem1.4 ファミリーをテンプレートとして使用して、14 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 myMem14  \
    --cache-parameter-group-family memcached1.4 \
    --description "My first parameter group"
```

Windows の場合:

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myMem14  ^
    --cache-parameter-group-family memcached1.4 ^
    --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": "myMem14",
        "CacheParameterGroupFamily": "memcached1.4",
        "Description": "My first  parameter group"
    }
}
```

Example

次の例では、redismyRed2.8 ファミリーをテンプレートとして使用して、28 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、「ElastiCache パラメータグループの変更」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。

ElastiCache パラメータグループの作成 (ElastiCache API)

を使用してパラメータグループを作成するには ElastiCache API、これらのパラメータで 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。
  - ASCII 文字で始まる必要があります。
  - 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、memcached1.4 と指定します。

- 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 と指定します。
- 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、memcachedmyMem1.4 ファミリーをテンプレートとして使用して、14 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=memcached1.4
   &CacheParameterGroupName=myMem14
   &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>myMem14</CacheParameterGroupName>
      <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
      <Description>My first  parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

Example

次の例では、redismyRed2.8 ファミリーをテンプレートとして使用して、28 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、「ElastiCache パラメータグループの変更」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。

名前による ElastiCache パラメータグループの一覧表示

パラメータグループは、 ElastiCache コンソール、 AWS CLI、または ElastiCache を使用して一覧表示できますAPI。

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ています。

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. 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 [パラメータグループ] を選択します。

名前による ElastiCache パラメータグループの一覧表示 (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 AWS CLI、 コマンド を使用しますdescribe-cache-parameter-groups。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 myMem14 を一覧表示します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myMem14
```

Windows の場合:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myMem14
```

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": "myMem14",
         "CacheParameterGroupFamily": "memcached1.4",
         "Description": "My first parameter group"
     }
    ]
}
```

Example

次のサンプルコードは、パラメータグループ myRed28 を一覧表示します。

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 OSS エンジンバージョン myRed5.0.6 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 56 を一覧表示します。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、パラメータグループの名前、ファミリー、 isGlobal 説明を一覧表示して、次のようなようになります。

```
{
    "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、および redis5.6 の場合、パラメータグループがグローバルデータストア (isGlobal) の一部であるかどうかをパラメータグループごとに一覧表示して、次のように表示されます。

```
{
    "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。

名前による ElastiCache パラメータグループの一覧表示 (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 ElastiCache API、`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は、パラメータグループ myMem14 を一覧表示します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myMem14
   &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、各グループパラメータの名前の一覧、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>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 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レスポンスは、名前、ファミリー、説明、およびパラメータグループがグローバルデータストア (isGlobal) に属する場合は redis5.6 の場合、パラメータグループごとにリストされます。

```
<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 を一覧表示します。

```
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 を一覧表示します。

```
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レスポンスは、名前、ファミリー、説明、および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一覧表示して、次のように表示されますisGlobal。

```
<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。

ElastiCache パラメータグループの値を一覧表示する

コンソール、、 AWS CLIまたは を使用して ElastiCache、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 ElastiCache API。

ElastiCache 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その値を一覧表示する方法を示しています。

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一覧表示するには ElastiCache

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. 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 [パラメータグループ] 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に必要がある場合もあります。

パラメータグループの値を一覧表示する (AWS CLI)

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 AWS CLI、 コマンド を使用しますdescribe-cache-parameters。

Example

次のサンプルコードは、パラメータグループ myMem14 のすべての Memcached パラメータとその値を一覧表示します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myMem14
```

Windows の場合:

```
aws elasticache describe-cache-parameters ^
```

```
    --cache-parameter-group-name myMem14
```

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その値を一覧表示します。

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)

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 ElastiCache API、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 myMem14 のすべての Memcached パラメータを一覧表示します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myMem14
   &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>
```

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表示します。

```
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。

ElastiCache パラメータグループの変更

> **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タイミングの詳細については、Valkey パラメータと Redis OSSパラメータ「」および「」を参照してくださいMemcached 固有のパラメータ。

パラメータグループを変更する (コンソール)

次の手順は、 ElastiCache コンソールを使用してcluster-enabledパラメータの値を変更する方法を示しています。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の値を変更するには

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. 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 [パラメータグループ] 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、[パラメータの編集] を選択します。
5. [パラメータグループの編集] 画面で、binding_protocol パラメータが見つかるまで、左右の矢印を使用してスクロールしてから、[値] 列に ascii と入力します。
6. [Save Changes] を選択します。

7. Memcached については、変更したパラメータの名前を確認するには、「」を参照してくださいMemcached 固有のパラメータ。再起動後にパラメータを変更する場合は、このパラメータグループを使用するクラスターを再起動します。詳細については、「クラスターの再起動」を参照してください。

8. Valkey と Redis ではOSS、変更したパラメータの名前を確認するには、「」を参照してくださいValkey パラメータと Redis OSSパラメータ。Valkey または Redis OSS (クラスターモードが無効) クラスターがあり、次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。

   > Valkey または Redis OSS (クラスターモードが有効) パラメータの変更
   >
   > Valkey または Redis OSS (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、次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. Redis を使用すると、クラスターの手動バックアップを再試行できます。「手動バックアップの取得」を参照してください。
   > 2. 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)

を使用してパラメータの値を変更するには AWS CLI、 コマンド を使用しますmodify-cache-parameter-group。

Example

Memcached では、変更するパラメータの名前と許可された値を見つけるには、「」を参照してください。 Memcached 固有のパラメータ

次のサンプルコードでは、パラメータグループ myMem14 で [chunk_size] と [chunk_size_growth_fact] の 2 つのパラメータの値を設定します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myMem14 \
    --parameter-name-values \
        ParameterName=chunk_size,ParameterValue=96 \
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

Windows の場合:

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name myMem14 ^
    --parameter-name-values ^
        ParameterName=chunk_size,ParameterValue=96 ^
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

このコマンドの出力は次のようになります。

```
{
    "CacheParameterGroupName": "myMem14"
}
```

Example

Valkey と Redis ではOSS、変更するパラメータの名前と許可された値を確認するには、「」を参照してください。 Valkey パラメータと Redis OSSパラメータ

次のサンプルコードは、2 つのパラメータの値を設定し、パラメータグループ でreserved-memory-percentクラスターを有効にしますmyredis32-on-30。パラメータグループを Valkey または Redis

30 (クラスターモードが有効) クラスター OSS (レプリケーショングループ) で使用できるyesように、 reserved-memory-percentを (30%) に設定し、クラスターを有効にして を に設定します。

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、「Valkey パラメータと Redis OSSパラメータ」を参照してください。

Valkey または Redis OSS (クラスターモードが無効) クラスターがあり、次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。

Valkey または Redis OSS (クラスターモードが有効) パラメータの変更

Valkey または Redis OSS (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、次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. 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)

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 ElastiCache API、`ModifyCacheParameterGroup`アクションを使用します。

Example

Memcached では、変更するパラメータの名前と許可された値を見つけるには、「」を参照してください。 Memcached 固有のパラメータ

次のサンプルコードでは、パラメータグループ `myMem14` で [chunk_size] と [chunk_size_growth_fact] の 2 つのパラメータの値を設定します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myMem14
   &ParameterNameValues.member.1.ParameterName=chunk_size
   &ParameterNameValues.member.1.ParameterValue=96
   &ParameterNameValues.member.2.ParameterName=chunk_size_growth_fact
   &ParameterNameValues.member.2.ParameterValue=1.5
```

```
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Example

Valkey と Redis ではOSS、変更するパラメータの名前と許可された値を確認するには、「」を参照してください。 Valkey パラメータと Redis OSSパラメータ

次のサンプルコードは、2 つのパラメータの値を設定し、パラメータグループ でreserved-memory-percentクラスターを有効にしますmyredis32-on-30。パラメータグループを Valkey または Redis 30 (クラスターモードが有効) クラスター OSS (レプリケーショングループ) で使用できるyesように、 reserved-memory-percentを (30%) に設定し、クラスターを有効にして を に設定します。

```
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。

Valkey または Redis OSS (クラスターモードが無効) クラスターがあり、次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。

Valkey または Redis OSS (クラスターモードが有効) パラメータの変更

Valkey または Redis OSS (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、次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. クラスターを削除します。「でのクラスターの削除 ElastiCache」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。

ElastiCache パラメータグループの削除

ElastiCache コンソール、、または を使用して AWS CLI、カスタムパラメータグループを削除できますElastiCache API。

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)

次の手順は、コンソールを使用して ElastiCacheパラメータグループを削除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. 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 [パラメータグループ] 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. [削除] を選択します。

   パラメータグループの削除の確認画面が表示されます。
5. パラメータグループを削除するには、[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、キャンセルを選択します。

パラメータグループを削除する (AWS CLI)

を使用してパラメータグループを削除するには AWS CLI、 コマンド を使用しますdelete-cache-parameter-group。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 個の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)

を使用してパラメータグループを削除するには ElastiCache API、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

Memcached では、次のサンプルコードは myMem14 個の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=myMem14
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Example

次のサンプルコードは、myRed28 個の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。

エンジン固有のパラメータ

Valkey と Redis OSS

Valkey 7.2 パラメータは Redis 7 OSS パラメータと同じです。

Valkey または Redis OSSクラスターにパラメータグループを指定しない場合、エンジンバージョンに適したデフォルトのパラメータグループが使用されます。デフォルトのパラメータグループのパラメータの値を変更することはできません。しかし、カスタムパラメータグループを作成し、いつでもクラスターに割り当てることはできます。ただし、条件付きで変更可能なパラメータの値が両方のパラメータグループで同じである場合に限ります。詳細については、「ElastiCache パラメータグループの作成」を参照してください。

トピック

- Valkey パラメータと Redis OSSパラメータ
- Memcached 固有のパラメータ

Valkey パラメータと Redis OSSパラメータ

トピック

Valkey 7.2 および Redis 7 OSS 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 OSS 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` – Valkey または Redis OSS (クラスターモードが無効) クラスターとレプリケーショングループには、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使用します。
- `default.redis7.cluster.on` – Valkey または Redis OSS (クラスターモードが有効) クラスターとレプリケーショングループには、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使用します。

Redis 7 OSS 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
cluster-allow-pubsubshard-when-down	許可される値: yes、no デフォルト: yes 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	デフォルトの [yes] 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。
cluster-preferred-endpoint-type	許可される値: ip、tls-dynamic デフォルト: tls-dynamic 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	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 のエンドポイントフィールドを制御しますCLUSTER SHARDS。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 tls-dynamic に設定されている場合、ノードは が有効なときに encryption-in-transitホスト名をアドバタイズし、それ以外の場合は ip アドレスをアドバタイズします。
latency-tracking	許可される値: yes、no デフォルト: no タイプ: 文字列 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	[yes] に設定すると、コマンドごとのレイテンシーが追跡され、INFO レイテンシー統計コマンドを使用してパーセンタイル分布をエクスポートし、LATENCY コマンドを使用して累積レイテンシー分布 (ヒストグラム) をエクスポートできます。
hash-max-listpack-entries	許可される値: 0+ デフォルト: 512	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。

名前	詳細	説明
	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	
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 OSS で変更されたパラメータは次のとおりです。

名前	詳細	説明
activerehashing	変更可能: no。Redis OSS7 では、このパラメータはデフォルトで非表示になり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 OSS 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 OSS 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 OSS 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 – Valkey または Redis OSS (クラスターモードが無効) クラスターとレプリケーショングループには、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使用します。
- default.redis6.x.cluster.on – Valkey または Redis OSS (クラスターモードが有効) クラスターとレプリケーショングループには、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使用します。

> **Note**
>
> Redis OSS エンジンバージョン 6.2 では、 で使用するために r6gd ノードファミリーが導入されたときのデータ階層化 ElastiCache、r6gd ノードタイプでは、noeviction 、volatile-lru および allkeys-lru max-memory ポリシーのみがサポートされます。

詳細については、「ElastiCache (Redis OSS) バージョン 6.2 (拡張) 」および「ElastiCache (Redis OSS) バージョン 6.0 (拡張) 」を参照してください。

Redis 6.x OSS で追加されたパラメータは次のとおりです。

詳細	説明	
acl-pubsub-default (added in 6.2)	許可される値: resetchannels 、allchannels デフォルト: allchannels タイプ: 文字列 変更可能: はい 変更が有効になります。クラスターに関連付けられた既存の Redis OSS ユーザーには、引き続き既存のアクセス許可が付与されます。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 OSS ユーザーを更新します。	このクラスターにデプロイされたACLユーザーのデフォルトの pubsub チャネルアクセス許可。
cluster-allow-reads-when-down (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	はいに設定すると、ノードがプライマリのクォーラムに到達できない場合でも、Redis OSS (クラスターモードが有効) レプリケーショングループは読み取りコマンドの処理を続行します。 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。
tracking-table-max-keys (added in 6.0)	デフォルト: 1,000,000 タイプ: 数値 変更可能: はい	クライアント側のキャッシュを支援するために、Redis 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OSSをサポートします。

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	追跡されたキーが変更されると、無効化メッセージがすべてのクライアントに送信され、キャッシュされた値が無効になったことが通知されます。この値により、このテーブルの上限を指定できます。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。

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	Redis は、2 つのメカニズムによって存続する時間を超えたキーOSS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 が定期的なジョブの項目の有効期限が切れるためにOSS使用する労力の量を定義します。 デフォルト値の 1 では、期限切れのキーの 10% 以上をメモリに残さないようにします。また、合計メモリの 25% 以上を消費しないようにし、システムにレイテンシーを追加しようとします。この値を最大 10 まで増やすと、キーの期限切れに費やす労力を増やすことができます。トレードオフは高くなりCPU、レイテンシーも高くなる可能性があります。メモリ使用率が高く、CPU使用率の増加を許容できる場合を除き、値は 1 にすることをお勧めします。
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	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。

Redis 6.x OSS 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-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

Redis 5.0.3 OSS 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 OSS のデフォルトパラメータグループ

- default.redis5.0 – Valkey または Redis OSS (クラスターモードが無効) クラスターとレプリケーショングループには、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使用します。
- default.redis5.0.cluster.on – Valkey または Redis OSS (クラスターモードが有効) クラスターとレプリケーショングループには、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使用します。

Redis 5.0.3 OSS で追加されたパラメータ

名前	詳細	説明
rename-commands	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 OSS コマンドのスペース区切りリスト。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 BRPOPLPUSH 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 SUNION SUNIONSTORE 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 XRANGE XREVRANGE XLEN XREAD XGROUP XREADGROUP XACK XCLAIM XPENDING GEORADIUS_RO GEORADIUSBYMEMBER_RO LOLWUT XSETID SUBSTR

詳細については、「ElastiCache (Redis OSS) バージョン 5.0.6 (拡張)」を参照してください。

Redis 5.0.0 OSS 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 OSS のデフォルトパラメータグループ

- `default.redis5.0` – Valkey または Redis OSS (クラスターモードが無効) クラスターとレプリケーショングループには、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使用します。
- `default.redis5.0.cluster.on` – Valkey または Redis OSS (クラスターモードが有効) クラスターとレプリケーショングループには、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使用します。

Redis 5.0 OSS で追加されたパラメータ

名前	詳細	説明
`stream-node-max-bytes`	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 OSS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 の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 OSS の新機能」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 OSS で名前が変更されたパラメータ

名前	詳細	説明
`replica-lazy-flush`	デフォルト: はい タイプ: ブール値 変更可能: いいえ 以前の名前 ： slave-lazy-flush	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`	デフォルト: 値については、「Redis OSS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 以前の名前: client-output-buffer-limit-slave-hard-limit	Redis OSS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すると、クライアントは切断されます。
`client-output-buffer-limit-replica-soft-limit`	デフォルト: 値については、「Redis OSS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 以前の名前: client-output-buffer-limit-slave-soft-limit	Redis OSS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すると、クライアントは切断されますが、この条件が に続く場合にのみ切断されます`client-output-buffer-limit-replica-soft-seconds` 。

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 以前の名前: client-output-buffer-limit-slave-soft-seconds	Redis OSS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も長い間client-output-buffer-limit-replica-soft-limit バイトのままの場合、クライアントは切断されます。
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 のリードレプリカが独自のリードレプリカを持つOSS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 min-replicas-max-lag が 0 の場合、レプリカが使用できない場合でも、プライマリノードは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	デフォルト: 10 タイプ: 整数 変更可能: はい 以前の名前： min-slaves-max-lag 変更の適用: 即時	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-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。 このパラメータまたは min-replicas-to-write が 0 の場合、レプリカが使用できない場合でも、プライマリノードは常に書き込みリクエストを受け入れます。
close-on-replica-write	デフォルト: はい タイプ: ブール値 変更可能: はい 以前の名前： close-on-slave-write 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。

Redis 5.0 OSS で削除されたパラメータ

名前	詳細	説明
repl-timeout	デフォルト: 60 変更可能: いいえ	パラメータはこのバージョンでは使用できません。

Redis 4.0.10 OSS パラメータの変更

パラメータグループファミリー: redis4.0

Redis 4.0.x OSS のデフォルトパラメータグループ

- `default.redis4.0` – Valkey または Redis OSS (クラスターモードが無効) クラスターとレプリケーショングループには、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使用します。
- `default.redis4.0.cluster.on` – Valkey または Redis OSS (クラスターモードが有効) クラスターとレプリケーショングループには、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使用します。

Redis 4.0.10 OSS で変更されたパラメータ

名前	詳細	説明
`maxmemory-policy`	許可される値: `allkeys-lru` 、`volatile-lru` 、**`allkeys-lfu`** 、**`volatile-lfu`** 、`allkeys-random` 、`volatile-random` 、`volatile-ttl` 、`noeviction` デフォルト: volatile-lru タイプ: 文字列 変更可能: はい 変更の反映: 即時	`maxmemory-policy` 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`allkeys-lfu` 、2 つの新しい許可された値が追加されました。これは、おおよそのを使用してすべてのキーをエビクトしLFU、は`volatile-lfu` 、期限切れセットを持つキーLFU間でおおよその を使用してエビクトします。バージョン 6.2 では、データ階層化で使用するために r6gd ノードファミリーが導入された場合、`noeviction` 、`volatile-lru` および `allkeys-lru` max-memory ポリシーのみが r6gd ノードタイプでサポートされます。

Redis 4.0.10 OSS 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		
`lazyfree-lazy-eviction`	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 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	期限切れのキーで、非同期削除を実行します。
lazyfree-lazy-server-del	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	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。
slave-lazy-flush	許可される値: 該当なし 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	スレーブの同期中に非同期 flushDB を実行します。
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	デフラグの最小労力をCPUパーセンテージで表します。
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	デフラグの最大労力CPUの割合。
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 OSS 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 (Redis OSS) 3.2.10 追加のパラメータはサポートされていません。

Redis 3.2.6 OSS 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 OSS では、追加のパラメータはサポートされていません。

Redis 3.2.4 OSS 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 以降では、2 OSS 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 OSS3.2.4 を実行するときに、Valkey または Redis OSS (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 OSS3.2.4 の追加機能を使用する場合は、このパラメータグループまたはそこから派生するパラメータグループを指定します。
- `default.redis3.2.cluster.on` – Valkey または Redis OSS (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ときに、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 OSS の新しいパラメータ

- Redis 3.2.4 OSS で変更されたパラメータ (拡張)

Redis 3.2.4 OSS 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 OSS3.2.4 では、次の追加パラメータがサポートされています。

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	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。 • -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	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。

名前	詳細	説明
		• 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	これがクラスターモード (はい) の Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループか、クラスターモード以外の (いいえ) Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループかを示します。クラスターモードの Valkey または Redis OSS (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 個のノードグループにデータを分割できます。 * Redis 3.2.x OSS には 2 つのデフォルトパラメータグループがあります。 • `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	に設定するとyes、クラスターモードの Valkey または Redis OSS (クラスターモードが有効) ノードは、少なくとも 1 つのハッシュスロットが検出できない (使用可能なノードがサービスを提供していない) ことが検出されると、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。 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。

名前	詳細	説明
hll-sparse-max-bytes	デフォルト: 3000 タイプ: 整数 変更可能: はい 変更の適用: 即時	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。 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。 スパースエンコーディングによる O(N) という、速度を落とさずにスペース効率の高いエンコーディングの利点を得るにはPFADD、約 3000 の値をお勧めします。CPU が懸念ではないがスペースが の場合、値は約 10,000 に引き上げることができ、データセットはカーディナリティ HyperLogLogs が 0 ~ 15,000 の範囲の多くの で構成されます。

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	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、Redis OSS データフットプリントはノードのすべてのメモリを消費するまで増加します。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。メモリを予約することで、ページングの量を減らすために、Redis 以外のOSS目的で使用可能なメモリの一部を脇に置くことができます。 このパラメータは に固有であり ElastiCache、標準の Redis OSSディストリビューションの一部ではありません。 詳細については、「reserved-memory 」および「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

Redis 3.2.4 OSS で変更されたパラメータ (拡張)

パラメータグループファミリー: redis3.2

Redis 3.2.4 OSS では、次のパラメータが変更されました。

名前	詳細	変更
activerehashing	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 Redis OSSバージョンからアップグレードする場合は、まずappendonly 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 Redis OSSバージョンからアップグレードする場合は、まずappendfsync 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 OSS 2.8.24 (拡張) でパラメータを追加

パラメータグループファミリー: redis2.8

Redis 2.8.24 OSS では、追加のパラメータはサポートされていません。

Redis OSS 2.8.23 (拡張) でパラメータを追加

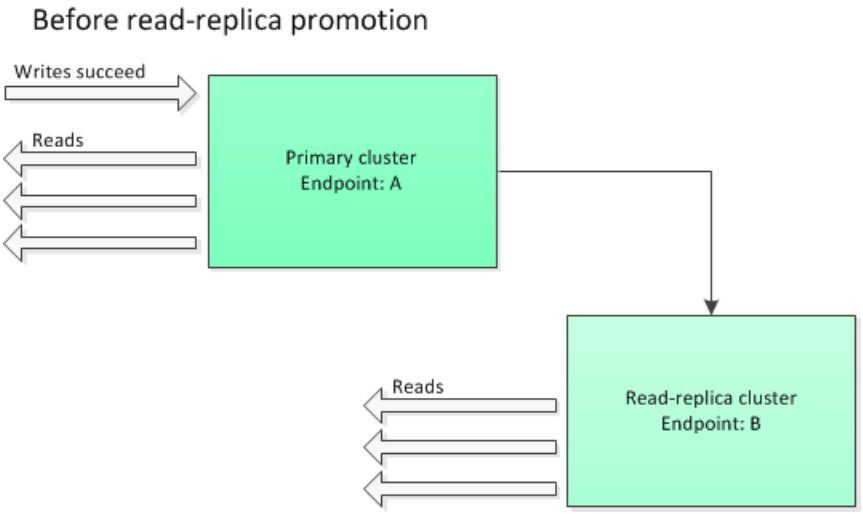
パラメータグループファミリー: redis2.8

Redis OSS2.8.23 では、次の追加パラメータがサポートされています。

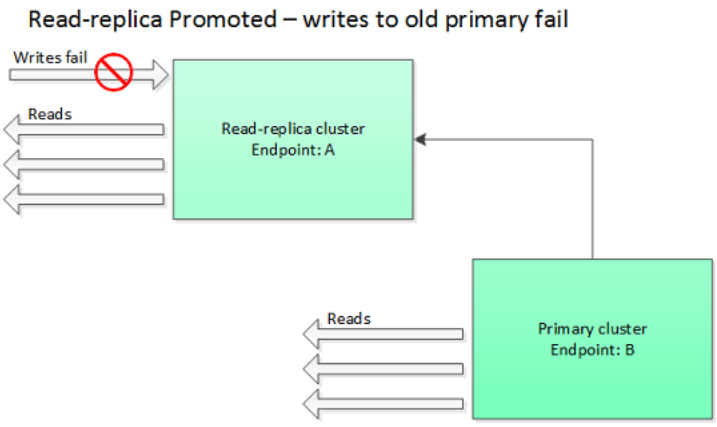
名前	詳細	説明
close-on-slave-write	デフォルト: はい タイプ: 文字列 (はい/いいえ) 変更可能: はい 変更の適用: 即時	有効にした場合、読み取り専用レプリカに書き込もうとするクライアントの接続は切断されます。

の close-on-slave-write仕組み

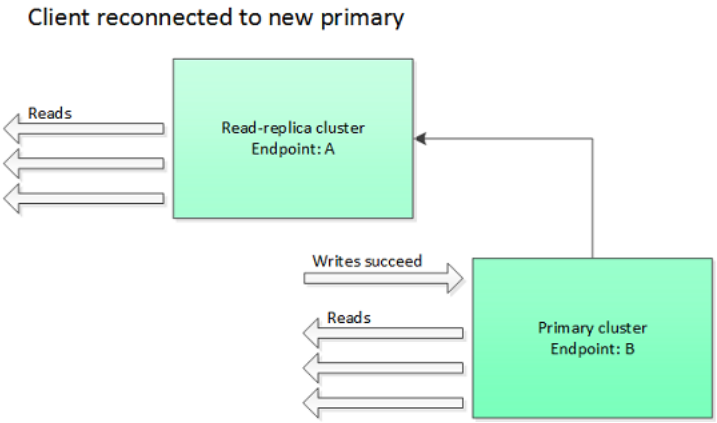
close-on-slave-write パラメータは Amazon によって導入され ElastiCache 、リードレプリカをプライマリに昇格させることにより、プライマリノードとリードレプリカノードスワップロールのクラスターの応答をより詳細に制御できます。



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OSS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テーブルをチェックして、エンドポイント B となるプライマリエンドポイントに再接続する必要があります。



を無効にできる場合 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になります。それでは、なぜ `close-on-replica-write` を無効にするのでしょう。

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 OSS 2.8.22 (拡張) でパラメータを追加

パラメータグループファミリー: redis2.8

Redis 2.8.22 OSS では、追加のパラメータはサポートされていません。

> **Important**
>
> - Redis OSSバージョン 2.8.22 以降、 `repl-backlog-size`はプライマリクラスターとレプリカクラスターに適用されます。
> - Redis OSSバージョン 2.8.22 以降、 `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 されます`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 OSS 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 OSS では、追加のパラメータはサポートされていません。

Redis 2.8.19 OSS 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 OSS では、追加のパラメータはサポートされていません。

Redis 2.8.6 OSS 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 OSS2.8.6 では、次の追加パラメータがサポートされています。

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	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。 このパラメータまたは min-slaves-to-write が 0 の場合、レプリカが使用できない場合でも、プライマリノードは常に書き込みリクエストを受け入れます。
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	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 min-slaves-max-lag が 0 の場合、レプリカが使用できない場合でも、プライマリノードは常に書き込みリクエストを受け入れます。
notify-keyspace-events	デフォルト: (空の文字列) タイプ: 文字列	Redis がクライアントに通知OSSできるキースペースイベントのタ

名前	詳細	説明
	変更可能: はい 変更の適用: 即時	イプ。各イベントタイプは 1 文字で表されます。 • K — Keyspace イベント、プレフィックス __keyspace@<db>__ を付けて発行 • E — Key-event イベント、プレフィックス __keyevent@<db>__ を付けて発行 • g — 、、 などの一般的な非固有コマンドDELEXPIRERENAME。 • $ — 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 — g$lshzxe のエイリアス

名前	詳細	説明
		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。例えば、 は Redis がすべてのイベントタイプの通知を発行OSSできるAKEことを意味します。 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。使用しようとすると、エラーメッセージが表示されます。 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。

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	プライマリノードバックログバッファーのサイズ (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。 このパラメータの最小値は 16384 です。 **Note** Redis OSS2.8.22 以降、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。

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	プライマリノードがバックログバッファーを保持する秒数。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ーを解放します。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。 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ーは解放されません。
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	次のタイムアウト時間 (秒単位) を表します。 • 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 OSS パラメータ

パラメータグループファミリー: redis2.6

Redis OSS 2.6.13 は、 でOSS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 OSS パラメータを示しています。

名前	詳細	説明
activerehashing	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	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。メインハッシュテーブルは 1 秒あたり 10 回リハッシュされます。各リハッシュオペレーションには 1 ミリ秒のCPU時間がかかります。 パラメータグループを作成するとき、この値を設定します。クラスターに新しいパラメータグループを割り当てるとき、この値は以前のパラメータグループと新しいパラメータグループで一致している必要があります。
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	Redis の追加のみのファイル機能 () を有効または無効にしますAOF。AOF は、キャッシュ内のデータを変更する Redis OSS コマンドをキャプチャし、特定のノード障害から復旧するために使用されます。 デフォルト値は ではありません。つまり、AOFはオフになっています。このパラメータをはいに設定して を有効にしますAOF。 詳細については、「障害の軽減」を参照してください。 **Note** 追加専用ファイル (AOF) は cache.t1.micro および cache.t2.* ノードではサポートされていません。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。

名前	詳細	説明
		Note マルチ AZ レプリケーショングループでは、 AOF は使用できません。
appendfsync	デフォルト: everysec タイプ: 文字列 変更可能: はい 変更の適用: 即時	appendonly がはいに設定されている場合、はAOF出力バッファがディスクに書き込まれる頻度を制御します。 • no — バッファーは必要に応じてディスクにフラッシュされます。 • everysec — バッファーは 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	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ーが指定されたバイト数に達した場合、クライアントの接続が切断されます。デフォルトは 0 です (ハード制限なし)。
client-output-buffer-limit-normal-soft-limit	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ーが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-normal-soft-seconds の間存続した場合に限ります。デフォルトは 0 です (ソフト制限なし)。

名前	詳細	説明
client-output-buffer-limit-normal-soft-seconds	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ーが、この秒数より長い時間 client-output-buffer-limit-normal-soft-limit バイトのままの場合、クライアントの接続が切断されます。デフォルトは 0 です (時間制限なし)。
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 のOSSパブリッシュ/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すると、クライアントは切断されます。
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 のOSSパブリッシュ/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すると、クライアントは切断されますが、この条件が に続く場合にのみ切断されますclient-output-buffer-limit-pubsub-soft-seconds 。
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 のOSSパブリッシュ/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も長くclient-output-buffer-limit-pubsub-soft-limit バイトに留まると、クライアントは切断されます。

名前	詳細	説明
client-output-buffer-limit-slave-hard-limit	デフォルト: 値については、「Redis OSS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 OSS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すると、クライアントは切断されます。
client-output-buffer-limit-slave-soft-limit	デフォルト: 値については、「Redis OSS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 OSS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すると、クライアントは切断されますが、この条件が に続く場合にのみ切断されますclient-output-buffer-limit-slave-soft-seconds 。
client-output-buffer-limit-slave-soft-seconds	デフォルト: 60 タイプ: 整数 変更可能: いいえ	Redis OSS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も長くclient-output-buffer-limit-slave-soft-limit バイトに留まると、クライアントは切断されます。
databases	デフォルト: 16 タイプ: 整数 変更可能: いいえ 変更の適用: 作成時	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。 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。

名前	詳細	説明
hash-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	ハッシュに使用されるメモリ量を決定します。エントリが指定された数より少ないハッシュは、領域を節約する特殊なエンコードを使用して格納されます。
hash-max-ziplist-value	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。

名前	詳細	説明
lua-time-limit	デフォルト: 5000 タイプ: 整数 変更可能: いいえ	Lua スクリプトの最大実行時間はミリ秒単位で、 ElastiCache はスクリプトを停止するアクションを実行します。 lua-time-limit を超えると、すべての Redis OSS コマンドは ____-BUSY という形式のエラーを返します。この状態は多くの重要な Redis OSSオペレーションに干渉する可能性があるため、 ElastiCache は最初にSCRIPTKILLコマンドを発行します。これが失敗した場合、 ElastiCache は強制的に Redis を再起動しますOSS。
maxclients この値は、明示的に指定されたインスタンスタイプを除くすべてのインスタンスタイプに適用されます。	デフォルト: 65000 タイプ: 整数 変更可能: いいえ	一度に接続できるクライアントの最大数。
	t2.medium デフォルト: 20000 タイプ: 整数 変更可能: いいえ	
	t2.small デフォルト: 20000 タイプ: 整数 変更可能: いいえ	
	t2.micro デフォルト: 20000 タイプ: 整数 変更可能: いいえ	

名前	詳細	説明
	t4g.micro デフォルト: 20000 タイプ: 整数 変更可能: いいえ	
	t3.medium デフォルト: 46000 タイプ: 整数 変更可能: いいえ	
	t3.small デフォルト: 46000 タイプ: 整数 変更可能: いいえ	
	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、LRU「キャッシュ OSS としての Valkey または Redis の使用」を参照してください。

名前	詳細	説明
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	(LRU) least-recently-usedと time-to-live (TTL) の計算の場合、このパラメータはチェックするキーのサンプルサイズを表します。デフォルトでは、Redis は 3 つのキーOSSを選択し、最近使用したキーを使用します。
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	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 OSS ノードはノードを消費するまで成長します maxmemory （「」を参照Redis OSSノードタイプ固有のパラメータ ）。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。メモリを予約することで、ページングの量を減らすために、Redis 以外のOSS目的で使用可能なメモリの一部を脇に置くことができます。 このパラメータは に固有であり ElastiCache、標準の Redis OSSディストリビューションの一部ではありません。 詳細については、「reserved-memory-percent 」および「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。
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	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。

名前	詳細	説明
slave-allow-chaining	デフォルト: いいえ タイプ: 文字列 変更可能: いいえ	Redis のリードレプリカが独自のリードレプリカを持つOSSことができるかどうかを決定します。
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 OSSスローログ機能によってログに記録されるコマンドの最大実行時間をマイクロ秒単位で指定します。
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 OSS スローログの最大長。
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。 **Important** このパラメータの一部の側面は、Redis OSSバージョン 3.2.4 で変更されました。「Redis 3.2.4 OSS で変更されたパラメータ (拡張)」を参照してください。

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。 • 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 OSS2.6.13 クラスターの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 OSS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	C lient-output-buffer-limit-slave-hard-limit	C lient-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	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-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	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-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	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-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	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-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	C lient-output-buffe r-limit-slave-hard- limit	C lient-output-buffe r-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**
>
> すべての現行世代のインスタンスタイプはVPC、デフォルトで Amazon Virtual Private Cloud に作成されます。
> T1 インスタンスはマルチ AZ をサポートしません。
> T1 インスタンスと T2 OSS インスタンスは Redis をサポートしていませんAOF。
> Redis OSS設定変数 `appendonly` および `appendfsync`は、Redis OSSバージョン 2.8.22 以降ではサポートされていません。

Memcached 固有のパラメータ

Memcached

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、「ElastiCache パラメータグループの作成」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	slab_reassign 、、lru_maintainer_thread lru_segmented 、およびmaxconns_fast コマンドを無効にするためのエイリアス。 Memcached 1.5 以降を使用する場合、 は hash_algorithm

名前	詳細	説明
		no_modern も に設定しますjenkins。 さらに、Memcached 1.5.10 を使用する場合、 inline_ascii_reponse は パラメータ によって制御されますparallelly 。つまり、 no_modern が無効になっている場合、 inline_ascii_reponse は無効になります。Memcached エンジン 1.5.16 以降、 inline_ascii_response パラメータは適用されなくなり、有効または無効にno_modern しても には影響しませんinline_ascii_reponse 。 no_modern が無効になっている場合は、slab_reassign 、lru_segmented 、、および lru_maintainer_thread が有効maxconns_fast WILLになります。slab_automove および hash_algorithm パラメータはSWITCHパラメータではないため、設定はパラメータグループの設定に基づいています。 を無効にno_modern して に戻す場合はmodern、このパラメータを無効にするようにカスタムパラメータグループを設定し、これら

名前	詳細	説明
		の変更を有効にするために再起動する必要があります。 Note 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、各リージョンの新しいElastiCache ユーザーによって自動的に取得されます。2021 年 8 月 20 日より前のリージョンの既存の ElastiCache ユーザーは、この新しい変更に対応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。
`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。`get`、ASCII `faster`セットの小さなスローダウン。

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_not_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	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。

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	

名前	詳細	説明
	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_automove 、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 変更可能: はい 変更の適用: 即時 ログは、watcher_logbuf_size および worker_logbuf_size 制限に達すると削除できます。	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 update が発生したときにユーザーはログを表示できます。
idle_timeout	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

名前	詳細	説明
track_sizes	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 track_sizes を行うと、stats sizes を実行せずに stats sizes_enable を実行できます。
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。

名前	詳細	説明
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す。slab_chunk_max より大きい項目は、複数のスラブに分割されます。
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 を実行できます。)
hash_algorithm	デフォルト: jenkins タイプ: 文字列	使用されるハッシュアルゴリズム。使用可能な値: murmur3 と jenkins。

名前	詳細	説明
	変更可能: はい 変更の適用: 起動時	

名前	詳細	説明
`lru_crawler`	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: 再起動後 **Note** 実行時に、コマンドラインから `lru_crawler` 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。	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 Crawl を起動する必要があります。 一時的に有効にするには、コマンドラインで `lru_crawler enable` を実行します。 `lru_crawler 1,3,5` はスラブクラス 1、3、5 をクロールし、freelist に追加する期限切れの項目を検索します。 値: 0、1 **Note** コマンドラインで `lru_crawler` 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、「ElastiCache パラメータグループの変更」を参照してください。

名前	詳細	説明
lru_maintainer	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	容量に達するLRUsと 間で項目をシャッフルするバックグラウンドスレッド。値: 0、1。
expirezero_does_not_evict	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	lru_maintainer 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。 **Warning** 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。 lru_maintainer を無視するよう設定できます。

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。2 の乗数で表されます。デフォルトは 16 (2^16)、つまり 65536 のキーです。

名前	説明
maxconns_fast	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。パラメータが 1 に設定されている場合、 はクライアントにエラー ElastiCache を送信し、接続をすぐに終了します。

名前	説明
slab_automove	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 に設定されている場合、 はスラブを自動的に移動するために、ゆっくりと保守的なアプローチElastiCache を取ります。2 に設定すると、立ち退きがあるたびにスラブが ElastiCache 激しく移動します。(このモードは、テスト目的以外では推奨されません)。

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、「ElastiCache パラメータグループの変更」を参照してください。
cas_disabled	デフォルト: 0 (false) 型: ブール値 変更可能: はい 変更の適用: 再起動後	1 (true) の場合、チェックして (CAS) オペレーションを設定すると無効になり、保存されたアイテムはCAS有効になっているものよりも 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: ブール値 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: ブール値 変更可能: いいえ	(1true) の場合、 ElastiCache は大きな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: ブール値 変更可能: いいえ	(true1) の場合、 ElastiCache はページ分割されたすべてのメモリを口ックします。
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: ブール値 変更可能: 変更の適用: 再起動後	(1true) の場合、 ElastiCache はコアファイルの制限を最大化します。

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 があるとします。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 の値を memcached_connections_overhead 13MB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) に作成されますVPC。

スケーリング ElastiCache

ニーズに合わせて ElastiCache キャッシュをスケーリングできます。サーバーレスキャッシュと独自設計のクラスターには、いくつかの異なるスケーリングオプションが用意されています。

ElastiCache サーバーレスのスケーリング

ElastiCache Serverless は、ワークロードトラフィックの増減に応じて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が制約されている場合、 ElastiCache サーバーレス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すること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、 のキャッシュによって消費され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制御するために、キャッシュデータストレージとキャッシュの ECPU/ 秒の両方で最大使用量を設定できます。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すると、キャッシュが上限に達すると、アプリケーションが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、 ElastiCache はLRUロジックを使用して (TTL) が Time-To-Live設定されたキャッシュ内のデータの削除を開始します。除外できるデータがない場合、追加データの書き込みリクエストには、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 はリクエストのス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、「 のドキュメント」を参照してください。

ElastiCache Serverless を使用したプリスケーリング

ElastiCache サーバーレスプリスケーリング

事前スケーリングは事前ウォームとも呼ばれ、 ElastiCache キャッシュでサポートされてい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起動から 1 分以内にログインが 5 倍増加すると予想した場合、この使用量の大幅な急増に備えてキャッシュを準備することができます。

ElastiCache コンソール、CLI、または を使用してプリスケーリングを実行できますAPI。 ElastiCache サーバーレスは 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。

プリスケーリングの仕組み

コンソール、、CLIまたは を介して ECPUs/秒またはデータストレージの最小制限が更新されるとAPI、その新しい制限が 1 時間以内に利用可能になります。 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場合、最大 90K ECPUs/秒をサポートします。 ElastiCache は 10～12 分ごとに 2 回 ECPUs/秒にできます。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。それ以外の場合、アプリケーションは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ECPUsあたりの新しい最小ストレージ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低くすると、更新はすぐに行われるため、 ElastiCache Serverless は新しい最小制限ですぐに計測を開始します。

> **Note**
>
> - 最小使用量制限を設定すると、実際の使用量が最小使用量制限を下回っていても、その制限に対して課金されます。ECPU またはデータストレージの使用量が最小使用制限を超えると、通常の料金が請求されます。例えば、最低使用制限を 100,000 ECPUs/秒に設定すると、その最低使用量を下回っても、1 時間あたり 1.224 USD (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 30KECPUs/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、90KECPUs/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 Valkey または Redis OSS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. 使用量制限 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

を使用してスケーリング制限を変更するにはCLI、 modify-serverless-cacheを使用しますAPI。

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}'
```

を使用したスケーリング制限の削除 CLI

を使用してスケーリング制限を削除するにはCLI、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}'
```

独自設計の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充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 をスケーリング ElastiCache できます。 を使用すると、使用している分に対してのみ料金を支払うことができます。 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。

> **Note**
>
> Valkey または Redis OSSクラスターが 1 つ以上のリージョンにレプリケートされている場合、それらのリージョンは順番にスケーリングされます。スケールアップすると、セカンダリリージョンが最初にスケーリングされ、次にプライマリリージョンがスケーリングされま

す。スケールダウンする場合、プライマリリージョンが最初に、次にセカンダリリージョンが続きます。
エンジンバージョンを更新する場合、順序はセカンダリリージョン、次にプライマリリージョンです。

トピック

Memcached のクラスターのスケーリング

次の表は、実行するスケーリングアクションに適したトピックを見つけるのに役立ちます。

Memcached クラスターのスケーリング

Memcached クラスターは 1～60 個のノードで構成されます。Memcached クラスターのスケールアウト/インはクラスターでのノードの追加/削除と同じくらい簡単です。

Memcached クラスターのすべてのノード間でデータを分割できるため、メモリのより大きいノードタイプにスケールアップすることはほとんど必要ありません。ただし、Memcached エンジンによってデータは永続的に保持されないため、別のノードタイプへのスケーリングを行う場合、新しいクラスターは、アプリケーションによって事前設定されない限り、最初は空の状態になります。

Memcached クラスターのスケーリング

アクション	トピック
スケールアウト	クラスターへのノードの追加
スケールイン	クラスターからノードの削除
ノードタイプの変更	Memcached の垂直スケーリング

Memcached クラスターは 1～60 個のノードで構成されます。Memcached クラスターのスケールアウト/インはクラスターでのノードの追加/削除と同じくらい簡単です。

Memcached クラスターのすべてのノード間でデータを分割できるため、メモリのより大きいノードタイプにスケールアップすることはほとんど必要ありません。ただし、Memcached エンジンによってデータは永続的に保持されないため、別のノードタイプへのスケーリングを行う場合、新しいクラスターは、アプリケーションによって事前設定されない限り、最初は空の状態になります。

トピック

- Memcached の水平スケーリング
- Memcached の垂直スケーリング

Memcached の水平スケーリング

Memcached エンジンでは、複数のノード間でのデータの分割がサポートされています。このため、Memcached クラスターの水平スケーリングは簡単です。Memcached クラスターには 1 ~ 60 個のノードを含めることができます。Memcached クラスターの水平スケーリングを行うには、ノードを追加または削除するだけです。

以下のトピックでは、ノードを追加したり削除したりして Memcached クラスターをスケーリングする方法について説明します。

- クラスターへのノードの追加
- クラスターからノードの削除

Memcached クラスターのノードの数を変更するたびに、正しいノードにマップできるようにキースペースの一部を再マッピングする必要があります。Memcached クラスターの負荷分散の詳細については、「効率的なロードバランシングのための ElastiCache クライアントの設定 (Memcached)」を参照してください。

Memcached クラスターで自動検出を使用する場合は、ノードを追加したり削除するたびに、アプリケーションのエンドポイントを変更する必要はありません。自動検出の詳細については、「クラスター内のノードを自動的に識別する (Memcached)」を参照してください。自動検出を使用しない場合は、Memcached クラスターのノード数を変更するたびに、アプリケーションのエンドポイントを更新する必要があります。

Memcached の垂直スケーリング

Memcached クラスターをスケールアップ/ダウンするときは、新しいクラスターを作成する必要があります。Memcached クラスターは、アプリケーションによって事前設定されない限り、最初は空の状態になります。

> **Important**
>
> より小さいノードタイプにスケールダウンする場合は、その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てください。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。

トピック

- Memcached の垂直スケーリング (コンソール)
- Memcached の垂直スケーリング (AWS CLI)
- Memcached の垂直スケーリング (ElastiCache API)

Memcached の垂直スケーリング (コンソール)

次の手順では、 ElastiCache コンソールを使用してクラスターを垂直にスケーリングする手順を説明します。

Memcached クラスターの垂直スケーリングを行うには (コンソール)

1. 新しいノードインスタンスタイプで新しいクラスターを作成します。詳細については、「Memcached クラスター (CLI) の作成 (コンソール)」を参照してください。
2. アプリケーションでは、新しいクラスターのエンドポイントにエンドポイントが更新されます。詳細については、「クラスターのエンドポイントの検索 (コンソール) (Memcached)」を参照してください。
3. 古いクラスターを削除します。詳細については、「Memcached での新しいノードの削除」を参照してください。

Memcached の垂直スケーリング (AWS CLI)

以下の手順では、 AWS CLIを使用した Memcached キャッシュクラスターの垂直スケーリングの手順について説明しています。

Memcached キャッシュクラスターの垂直スケーリングを行うには (AWS CLI)

1. 新しいノードインスタンスタイプで新しいキャッシュクラスターを作成します。詳細については、「 を使用したクラスターの作成CLI」を参照してください。

2. アプリケーションでは、新しいクラスターのエンドポイントにエンドポイントが更新されます。詳細については、「エンドポイントの検索 (AWS CLI)」を参照してください。
3. 古いキャッシュクラスターを削除します。詳細については、「AWS CLI を使用して ElastiCache クラスターを削除する」を参照してください。

Memcached の垂直スケーリング (ElastiCache API)

次の手順では、 を使用して Memcached キャッシュクラスターを垂直方向にスケーリングする手順を説明します ElastiCache API。

Memcached キャッシュクラスターを垂直にスケールするには (ElastiCache API)

1. 新しいノードインスタンスタイプで新しいキャッシュクラスターを作成します。詳細については、「Memcached 用のクラスターの作成 (ElastiCache API)」を参照してください
2. アプリケーションで、エンドポイントを新しいキャッシュ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、「エンドポイントの検出 (ElastiCache API)」を参照してください。
3. 古いキャッシュクラスターを削除します。詳細については、「の使用 ElastiCache API」を参照してください。

Valkey または Redis のクラスターのスケーリング OSS (クラスターモードが無効)

Valkey または Redis OSS (クラスターモードが無効) クラスターは、シャードが 0 の単一ノードクラスターでも、シャードが 1 の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。

トピック

- Valkey または Redis の単一ノードクラスターのスケーリング OSS (クラスターモードが無効)

Valkey または Redis OSSクラスターのスケーリング

アクション	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (クラスターモードが有効)
スケールイン	ElastiCache クラスターからノードを削除する	Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)
スケールアウト	クラスターへのノードの追加	Valkey または Redis のオンライン再シャーディング OSS (クラスターモードが有効)
ノードタイプの変更	より大きいノードタイプへ: • 単一ノードの Valkey または Redis OSSクラスターのスケールアップ • レプリカを使用した Valkey または Redis OSSクラスターのスケールアップ より小さいノードタイプへ: •	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	Valkey または Redis OSS (クラスターモードが無効)	Valkey または Redis OSS (クラスターモードが有効)
	単一ノードの Valkey または Redis OSSクラスターのスケールダウン • レプリカを使用した Valkey または Redis OSSクラスターのスケールダウン	
ノードグループの数の変更	Valkey または Redis OSS (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	Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)

目次

Valkey または Redis の単一ノードクラスターのスケーリング OSS (クラスターモードが無効)

Valkey または Redis OSS (クラスターモードが無効) ノードは、すべてのキャッシュのデータと Valkey または Redis オーバーOSSヘッドを含むのに十分な大きさである必要があります。Valkey クラスターまたは Redis OSS (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、垂直方向にスケールするか、より大きな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、より小さな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 スケールアッププロセスは、既存のデータを保持するために最善の努力をするように設計されており、成功した Valkey または Redis OSSレプリケーションが必要です。Valkey または Redis OSS (クラスターモードが無効) クラスターの場合、十分なメモリを Valkey または Redis で使用できるようにすることをお勧めしますOSS。

複数の Valkey または Redis OSS (クラスターモードが無効) クラスターにデータをパーティション化することはできません。ただし、クラスターの読み取り容量を増減するだけで済む場合は、レプリカノードを使用して Valkey または Redis OSS (クラスターモードが無効) クラスターを作成し、リードレプリカを追加または削除できます。単一ノードの Valkey または Redis OSSキャッシュクラスターをプライマリクラスターとして使用して、レプリカノードを持つ Valkey または Redis OSS (クラスターモードが無効) クラスターを作成するには、「」を参照してくださいValkey (クラスターモードが無効) クラスターの作成 (コンソール) 。

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。

レプリカを使用した Valkey または Redis OSS (クラスターモードが無効) クラスターは、読み取り容量を拡張できるだけでなく、その他のビジネス上の利点もあります。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 Valkey または Redis オーバーOSS ヘッドのメモリを確保している場合、スケーリングを開始する前に、新しいノードタイプに適切な量のメモリを予約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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

トピック

- 単一ノードの Valkey または Redis OSSクラスターのスケールアップ
- 単一ノードの Valkey または Redis OSSクラスターのスケールダウン

単一ノードの Valkey または Redis OSSクラスターのスケールアップ

単一ノードの Valkey クラスターまたは Redis OSSクラスターをスケールアップすると、ElastiCache はコンソール、、または を使用するかどうかにかかわらず AWS CLI、次のプロセス ElastiCache を実行します ElastiCache API。

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ー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した。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のと同じなので、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 エントリの更新中に、プライマリノードからの読み取りと書き込みが短時間 (数秒) 中断されます。
4. ElastiCache は古いキャッシュクラスターを削除します。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。

> **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。

次の表に示すように、次のメンテナンスウィンドウにエンジンアップグレードがスケジュールされている場合、Valkey または Redis のOSS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、「ElastiCache クラスター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 Valkey または Redis OSS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

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
	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。

- 即時適用チェックボックスをオフにして、次のメンテナンスウィンドウの Valkey または Redis OSSスケールアップオペレーションをスケジュールします (CLI を使用: `--no-apply-immediately`、 APIを使用: `ApplyImmediately=false`) 。
- 次のメンテナンスウィンドウ (またはそれ以降) が Valkey または Redis OSSスケールアップオペレーションを実行するまで待ちます。
- このキャッシュクラスターの変更に Valkey または Redis OSS エンジンのアップグレードを追加するときは、「すぐに適用」チェックボックスを選択します (CLI使用: `--apply-immediately`、API使用: `ApplyImmediately=true`) 。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、、または を使用して、単一ノードの Valkey AWS CLIまたは Redis OSS (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 ElastiCache API。

> **Important**
>
> パラメータグループが `reserved-memory`を使用して Valkey または Redis オーバーOSS ヘッドのメモリを確保している場合、スケーリングを開始する前に、新しいノードタイプに適切な量のメモリを予約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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

Valkey または Redis の単一ノードクラスターのスケールアップ OSS (クラスターモードが無効) (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単一ノードの Valkey または Redis OSS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、Valkey または Redis OSSクラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。

単一ノードの Valkey または Redis OSS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から、Valkey または Redis OSS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 (クラスター化された Valkey または Redis OSS エンジンではなく、Valkey または Redis OSS エンジンを実行している必要があります)。
4. Modify 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。
   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 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。

単一ノードの Valkey または Redis OSSキャッシュクラスターのスケールアップ (AWS CLI)

次の手順では、 を使用して単一ノードの Valkey または Redis OSS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ます AWS CLI。このプロセス中、Valkey または Redis OSSクラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。

単一ノードの Valkey または Redis OSS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. 次のパラメータを使用して `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。

   - `--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
   ```

   上記のコマンドからの出力は次のようになります (JSON 形式) 。

   ```
   {
        "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、以下を参照してください。list-allowed-node-type- 変更 リファレンス のAWS CLI 「」。

2. コマンドと次のパラメータを使用して AWS CLI modify-cache-cluster、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なノードタイプを指定する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
```

上記のコマンドからの出力は次のようになります (JSON 形式) 。

```
{
    "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、以下を参照してください。modify-cache-cluster AWS CLI リファレンス の「」。

3. を使用した場合は`--apply-immediately`、次のパラメータを使用して コマンドを使用して AWS CLI `describe-cache-clusters`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` – 単一ノードの Valkey または Redis OSS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、以下を参照してください。describe-cache-clusters リファレンス のAWS CLI「」。

単一ノードの Valkey または Redis OSSキャッシュクラスターのスケールアップ (ElastiCache API)

次の手順では、 を使用して単一ノードの Valkey または Redis OSS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ます ElastiCache API。このプロセス中、Valkey または Redis OSS クラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。

単一ノードの Valkey または Redis OSS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. 次のパラメータを使用して ElastiCache APIListAllowedNodeTypeModifications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決定します。

   - `CacheClusterId` – スケールアップする単一ノードの Valkey または Redis OSS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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」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、既存のキャッシュクラスターを変更し、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なノードタイプを指定します。

   - 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、前のステップの ListAllowedNodeTypeModificationsアクションによって返されるノードタイプの 1 つである必要があります。
   - 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-xl
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   詳細については、以下を参照してください。ModifyCacheCluster 「Amazon ElastiCache API リファレンス」の「」。

3. を使用した場合はApplyImmediately=true、次のパラメータを使用して ElastiCache APIDescribeCacheClustersアクションを使用して、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` – 単一ノードの Valkey または Redis OSS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、以下を参照してください。DescribeCacheClusters 「Amazon ElastiCache API リファレンス」の「」。

単一ノードの Valkey または Redis OSSクラスターのスケールダウン

以下のセクションでは、単一ノードの Valkey または Redis OSSクラスターを小さな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。新しい小さなノードタイプがすべてのデータや Valkey または Redis のOSSオーバーヘッドに対応できる大きさであることを確認することは、新しい Valkey または Redis OSSクラスターの長期的な成功にとって重要です。詳細については、「Valkey または Redis OSS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を確認する」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。

トピック

- 単一ノードの Valkey または Redis OSSクラスターのスケールダウン (コンソール)
- 単一ノードの Valkey または Redis OSSキャッシュクラスターのスケールダウン (AWS CLI)
- 単一ノードの Valkey または Redis OSSキャッシュクラスターのスケールダウン (ElastiCache API)

単一ノードの Valkey または Redis OSS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 コンソールを使用して、単一ノードの Valkey または Redis OSSクラスターを小さなノードタイプにスケールダウンする手順を説明します。

> **Important**
>
> パラメータグループが `reserved-memory`を使用して Valkey または Redis オーバーOSSヘッドのメモリを確保している場合、スケーリングを開始する前に、新しいノードタイプに適切な量のメモリを予約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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

単一ノードの Valkey または Redis OSS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 Valkey または Redis オーバーOSSヘッドのメモリを脇に置く場合は、新しいノードタイプに適切なメモリ量を脇に置く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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。
3. 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。
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、クラスター化された Valkey または Redis OSS エンジンではなく、Valkey または Redis OSS エンジンを実行している必要があります。
5. Modify 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 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。

単一ノードの Valkey または Redis OSS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、 を使用して単一ノードの Valkey または Redis OSS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 AWS CLI。

単一ノードの Valkey または Redis OSS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. 次のパラメータを使用して `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。

   - `--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
   ```

   上記のコマンドからの出力は次のようになります (JSON 形式) 。

   ```
   {
        "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、以下を参照してください。list-allowed-node-type- 変更 リファレンス のAWS CLI 「」。

2. コマンドと次のパラメータを使用して AWS CLI modify-cache-cluster、既存のキャッシュクラスターを変更し、スケールダウンするキャッシュクラスターと新しい小さなノードタイプを指定します。

   - --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-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
```

上記のコマンドからの出力は次のようになります (JSON 形式) 。

```
{
    "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、以下を参照してください。modify-cache-cluster AWS CLI リファレンス の「」。

3. を使用した場合は--apply-immediately、次のパラメータを使用して コマンドを使用して AWS CLI describe-cache-clusters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` – 単一ノードの Valkey または Redis OSS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、以下を参照してください。describe-cache-clusters リファレンス のAWS CLI「」。

単一ノードの Valkey または Redis OSS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、 を使用して単一ノードの Valkey または Redis OSS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 ElastiCache API。

単一ノードの Valkey または Redis OSS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. 次のパラメータを使用して ElastiCache APIListAllowedNodeTypeModifications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決定します。

   - `CacheClusterId` – スケールダウンする単一ノードの Valkey または Redis OSS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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」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、既存のキャッシュクラスターを変更し、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なノードタイプを指定します。

   - 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、前のステップの ListAllowedNodeTypeModificationsアクションによって返されるノードタイプの 1 つである必要があります。
   - 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、以下を参照してください。ModifyCacheCluster「Amazon ElastiCache API リファレンス」の「」を参照してください。

3. を使用した場合はApplyImmediately=true、次のパラメータを使用して ElastiCache APIDescribeCacheClustersアクションを使用して、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。

- `CacheClusterId` – 単一ノードの Valkey または Redis OSS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、以下を参照してください。DescribeCacheClusters 「Amazon ElastiCache API リファレンス」の「」を参照してください。

Valkey または Redis のレプリカノードのスケーリング OSS (クラスターモードが無効)

レプリカノードを持つ Valkey または Redis OSSクラスター (API/ ではレプリケーショングループと呼ばれますCLI) は、自動フェイルオーバーが有効になっているマルチ AZ を持つレプリケーションを介して高可用性を提供します。レプリカノードを持つクラスターは、最大 6 つの Valkey ノードまたは Redis OSSノードの論理コレクションであり、1 つのノードである Primary は読み取りリクエストと書き込みリクエストの両方を処理できます。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。Valkey または Redis OSS (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、Valkey または Redis OSS (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。Valkey または Redis OSS (クラスターモードが有効) クラスターは、最大 500 個のシャードにデータをパーティション化できます。

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 スケールアッププロセスは、既存のデータを保持するために最善の努力をするように設計されており、成功した Valkey または Redis OSSレプリケーションが必要です。レプリカを持つ Valkey または Redis OSSクラスターの場合、十分なメモリを Valkey または Redis で使用できるようにすることをお勧めしますOSS。

トピック

- レプリカを使用した Valkey または Redis OSSクラスターのスケールアップ
- レプリカを使用した Valkey または Redis OSS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

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: Valkey および Redis OSS クラスターモード無効と有効
- Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS

- Valkey または Redis OSS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を確認する

トピック

- レプリカを使用した Valkey または Redis OSSクラスターのスケールアップ
- レプリカを使用した Valkey または Redis OSS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

レプリカを使用した Valkey または Redis OSSクラスターのスケールアップ

Amazon ElastiCache は、Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをスケールアップするためのコンソールCLI、、およびAPIサポートを提供します。

スケールアッププロセスが開始されると、 は以下 ElastiCache を実行します。

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. 新しいノードを指すように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。Valkey 7.2 以降または Redis OSS5.0.5 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、自動フェイルオーバーが有効なクラスターをスケールできます。Redis OSSバージョン 4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。
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。

次の表に示すように、クラスターの次のメンテナンスウィンドウにエンジンアップグレードがスケジュールされている場合、Valkey または Redis OSS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 Valkey または Redis OSS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。

- 即時適用チェックボックスをオフにして、次のメンテナンスウィンドウの Valkey または Redis OSSスケールアップオペレーションをスケジュールします (CLI を使用: `--no-apply-immediately`、 APIを使用: `ApplyImmediately=false`) 。
- 次のメンテナンスウィンドウ (またはそれ以降) が Valkey または Redis OSSスケールアップオペレーションを実行するまで待ちます。
- 選択した Apply immediately チェックボックスを使用して、このキャッシュクラスターの変更に Valkey または Redis OSS エンジンのアップグレードを追加します (CLI使用: `--apply-immediately`、API使用: `ApplyImmediately=true`) 。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および を使用して、レプリカを使用して Valkey または Redis OSS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 ElastiCache API。

> **Important**
>
> パラメータグループが `reserved-memory`を使用して Valkey または Redis オーバーOSS ヘッドのメモリを確保している場合、スケーリングを開始する前に、新しいノードタイプに適切な量のメモリを予約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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

レプリカを使用した Valkey または Redis OSS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。

次のプロセスでは、 ElastiCache コンソールを使用して、レプリカを使用してクラスターを現在のノードタイプから新しいより大きな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 エントリの更新中に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が発生し、古いバージョンでは数秒未満のダウンタイムが発生する場合があります。

レプリカを使用して Valkey または Redis OSS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、Valkey クラスターまたは Redis OSS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、クラスター化された Valkey または Redis OSS エンジンではなく、Valkey または Redis OSS エンジンを実行している必要があります。
4. Modify 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 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。

Valkey または Redis OSS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。このプロセス中、 はDNSエントリ ElastiCache 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。Valkey 7.2 以降または Redis OSS5.0.5 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、自動フェイルオーバーが有効になっているクラスターをスケールできます。バージョン 4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。

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。

Valkey または Redis OSS 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. 次のパラメータを使用して `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。

   - `--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次のようになります (JSON 形式) 。

   ```
   {
        "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、以下を参照してください。list-allowed-node-type- 変更 AWS CLI リファレンスの「」。

2. 次のパラメータを指定して modify-replication-group コマンド AWS CLI 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、前のステップの 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 1 つ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-m3-2xl \
        --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-m3-2xl \
        --apply-immediately
   ```

   このコマンドからの出力は次のようになります (JSON 形式) 。

```
{
 "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、以下を参照してください。modify-replication-group AWS CLI リファレンス の「」。

3. --apply-immediately パラメータを使用する場合は、次のパラメータを使用して describe-replication-group コマンドを使用して AWS CLI 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。ステータスがまだ を変更している間、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、DNSエントリの更新中に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されることがあります。

   - --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、 リファレンスdescribe-replication-groupsの「」を参照してください。 AWS CLI

Valkey または Redis OSS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、 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なノードタイプにスケーリングします ElastiCache API。Valkey 7.2 以降または Redis 5.0.5 OSS 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、自動フェイルオーバーが有効になっているクラスターをスケールできます。バージョン Redis OSS4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。

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。

Valkey または Redis OSS 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. 次のパラメータを使用して、 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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」を参照してください。

2. `ModifyRedplicationGroup` ElastiCache API アクションと次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、前のステップの `ListAllowedNodeTypeModifications`アクションによって返されるインスタンスタイプの 1 つである必要があります。
   - `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-m3-2xl
    &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、以下を参照してください。ModifyReplicationGroup 「Amazon ElastiCache API リファレンス」の「」を参照してください。

3. を使用した場合はApplyImmediately=true、次のパラメータを使用して ElastiCache APIDescribeReplicationGroups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、以下を参照してください。DescribeReplicationGroups 「Amazon ElastiCache API リファレンス」の「」。

レプリカを使用した Valkey または Redis OSSクラスターのスケールダウン

以下のセクションでは、レプリカノードを使用して Valkey または Redis OSS (クラスターモードが無効) キャッシュクラスターを小さなノードタイプにスケール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、「Valkey または Redis OSS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を確認する」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 Valkey または Redis オーバーOSSヘッドのメモリを確保している場合、スケーリングを開始する前に、新しいノードタイプに適切な量のメモリを予約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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

レプリカを使用した Valkey または Redis OSSクラスターのスケールダウン (コンソール)

次のプロセスは、 ElastiCache コンソールを使用して、レプリカノードを持つ Valkey または Redis OSSクラスターを小さなノードタイプにスケーリングします。

レプリカノードを使用して Valkey または Redis OSS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 Valkey または Redis オーバーOSSヘッドのメモリを脇に置く場合は、新しいノードタイプに適切なメモリ量を脇に置く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、「Valkey と Redis の予約済みメモリの管理 OSS」を参照してください。

3. 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。
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、クラスター化された Valkey または Redis OSS エンジンではなく、Valkey または Redis OSS エンジンを実行している必要があります。
5. Modify 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 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。

Valkey または Redis OSS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。このプロセス中、 はDNSエントリ ElastiCache 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。Valkey 7.2 以上または Redis 5.0.5 OSS 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、自動フェイルオーバーが有効になっているクラスターをスケールできます。バージョン 4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。

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。

Valkey または Redis OSS 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 `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。

   - `--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次のようになります (JSON 形式) 。

   ```
   {
        "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、以下を参照してください。list-allowed-node-type- 変更 AWS CLI リファレンスの「」。

2. 次のパラメータを指定して modify-replication-group コマンド AWS CLI 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、前のステップの 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 1 つ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-m3-2xl \
        --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-m3-2xl \
        --apply-immediately
   ```

このコマンドからの出力は次のようになります (JSON 形式) 。

```
{"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、以下を参照してください。modify-replication-group リファレンス のAWS CLI「」を参照してください。

3. `--apply-immediately` パラメータを使用する場合は、次のパラメータを使用して `describe-replication-group` コマンドを使用して AWS CLI 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、「 リファレンスdescribe-replication-groups」の「」を参照してください。AWS CLI

Valkey または Redis OSS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、 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なノードタイプにスケーリングします ElastiCache API。このプロセス中、 はDNSエントリ ElastiCache 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。Valkey 7.2 以降または Redis OSS5.0.5 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、自動フェイルオーバーが有効になっているクラスターをスケールできます。Redis OSSバージョン 4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。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。

Valkey または Redis OSS 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. 次のパラメータを使用して、 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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」。

2. `ModifyRedplicationGroup` ElastiCache API アクションと次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、前のステップの ListAllowedNodeTypeModificationsアクションによって返されるインスタンスタイプの 1 つである必要があります。
- 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-m3-2xl
    &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、以下を参照してください。ModifyReplicationGroup 「Amazon ElastiCache API リファレンス」の「」。

3. を使用した場合はApplyImmediately=true、次のパラメータを使用して ElastiCache APIDescribeReplicationGroups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、以下を参照してください。DescribeReplicationGroups 「Amazon ElastiCache API リファレンス」の「」を参照してください。

読み込みキャパシティーの増加

読み取り容量を増やすには、Valkey または Redis レプリOSSケーショングループにリードレプリカ (最大 5 つ) を追加します。

ElastiCache コンソール、、または を使用して、Valkey AWS CLIまたは Redis OSSクラスターの読み取り容量をスケーリングできます ElastiCache API。詳細については、「Valkey または Redis のリードレプリカの追加 OSS (クラスターモードが無効)」を参照してください。

読み込みキャパシティーの削減

リードキャパシティーを減らすには、レプリカ (API/ ではレプリケーショングループと呼ばれます CLI) を使用して Valkey または Redis OSSクラスターから 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、「Valkey または Redis のリードレプリカの削除 OSS (クラスターモードが無効)」を参照してください。

Valkey または Redis でのクラスターのスケーリング OSS (クラスターモードが有効)

クラスターに対する需要が変化すると、Valkey または Redis OSS (クラスターモードが有効) クラスター内のシャードの数を変更することで、パフォーマンスを向上させたり、コストを削減したりできます。そのために、スケーリングプロセス中でもクラスターがリクエストを処理し続けることができる、オンライン水平スケーリングの使用をお勧めします。

クラスターを再スケーリングするかどうかの判断条件には、次のようなものがあります。

- メモリプレッシャー:

  クラスター内のノードがメモリプレッシャーを受けている場合、より多くのリソースがより効率よくデータを保存してリクエストを処理するようにスケールアウトできます。

  ノードがメモリ負荷を受けているかどうかはFreeableMemory、、SwapUsage、および のメトリクスをモニタリングすることで判断できますBytesUseForCache。
- CPU またはネットワークのボトルネック :

  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。

  レイテンシーとスループットレベル
  は、、CPUUtilization、、NetworkBytesInNetworkBytesOutCurrConnections、および のメトリクスをモニタリングすることでモニタリングできますNewConnections。
- 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、、、BytesUseForCacheCPUUtilizationNetworkBytesInNetworkB
  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確認できま
  すNewConnections。

パフォーマンスに対するスケーリングの影響

オフライン処理を使用してスケーリングすると、処理の大部分でクラスターがオフラインになるため、リクエストに対応できなくなります。オンラインメソッドを使用してスケーリングすると、スケーリングは大量の演算を行うオペレーションであるため、パフォーマンスがある程度低下します。

その場合でも、クラスターはスケーリングオペレーション全体を通してリクエストに対応しつづけます。発生する劣化の程度は、通常のCPU使用率とデータによって異なります。

Valkey クラスターまたは Redis OSS (クラスターモードが有効) クラスターをスケールするには、水平スケーリングと垂直スケーリングの 2 つの方法があります。

- 水平スケーリングでは、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。

クラスターのサイズとメモリ容量を縮小する場合は、スケールインまたはスケールダウンして、新しい設定にデータと Valkey または Redis オーバーOSSヘッド用の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、「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

- 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
    - Valkey または Redis OSSキャッシュクラスターのスケールアップ (コンソール)
    - Valkey または Redis OSSキャッシュクラスターのスケールアップ (AWS CLI)
    - Valkey または Redis OSS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
    - Valkey または Redis OSSキャッシュクラスターのスケールダウン (コンソール)
    - Valkey または Redis OSSキャッシュクラスターのスケールダウン (AWS CLI)
    - Valkey または Redis OSSキャッシュクラスターのスケールダウン (ElastiCache API)

Valkey または Redis のオフライン再シャーディング OSS (クラスターモードが有効)

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。オフラインで再シャードと再調整を行う場合は、レプリケーショングループのシャードの数を変更するだけでなく、次の操作を実行できます。

> **Note**
>
> データ階層化が有効になっている Valkey または Redis OSSクラスターでは、オフラインの再シャーディングはサポートされていません。詳細については、「のデータ階層化 ElastiCache」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、アベイラビリティー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 Valkey または Redis OSS (クラスターモードが有効) クラスターをオフラインで再設定するには

1. 既存の Valkey または Redis OSS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、「での接続エンドポイントの検索 ElastiCache」を参照してください。

Valkey または Redis のオンライン再シャーディング OSS (クラスターモードが有効)

ElastiCache Valkey 7.2 以降、または Redis OSS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リバランシングを使用すると、ダウンタイムなしで Valkey または Redis OSS (クラスターモードが有効) クラスター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 – Valkey または Redis (クラスターモードが有効) クラスター OSS (レプリケーショングループ) にシャード (ノードグループ) を追加して、読み取りおよび書き込み容量を増やします。

  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 – Valkey または Redis OSS (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、読み取り容量と書き込み容量を削減し、コストを削減します。
- 再調整 – Valkey または Redis OSS (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、シャード間で可能な限り均等に分散されるようにします。

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 オンラインのリシャーディングとリバランシングには以下の制限が適用されます。

- これらのプロセスには、Valkey 7.2 以降または Redis OSS 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、「のバージョン管理 ElastiCache」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。
- グローバルデータストア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 (複数可) に関数を自動的にレプリケートしません。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 Valkey 7.2 以降および Redis OSSバージョン 7 以降では、クラスターをスケールアウトすると、 ElastiCache は既存のノード (ランダムに選択) の 1 つにロードされた関数を新しいノード (複数可) に自動的にレプリケートします。アプリケーションが 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することをお勧めします。これにより、クラスターが異なるシャードで異なる関数定義に終わることがなくなります。

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 を使用して AWS Management Console、Valkey または Redis OSS (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ルまたは再調整できます ElastiCache API。

オンラインリシャーディングによるシャードの追加

AWS Management Console、 AWS CLI、または ElastiCache を使用して、Valkey または Redis OSS (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API。Valkey または Redis OSS (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 Valkey または Redis OSS (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。

Valkey または Redis OSS (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には

1. で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインから、Valkey クラスターまたは Redis OSSクラスター を選択します。
3. シャードを追加する Valkey または Redis OSS (クラスターモードが有効) クラスターの名前を見つけて選択します。

   > **Tip**
   >
   > Valkey または Redis OSS (クラスターモードが有効) は、モード列に Clustered Valkey または Clustered Redis OSS を表示します。

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. [アベイラビリティーゾーン] 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ー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、 を使用してシャードを追加して、Valkey または Redis OSS (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を説明します 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。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ー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、アベイラビリティーゾーンのリスト。`--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ーゾーンを選択します。

次の例では、 という名前の Valkey または Redis OSS (クラスターモードが有効) クラスター内の 4 つのシャードのキースペースを再設定しますmy-cluster。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ー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

オペレーションを使用して ElastiCache API、Valkey クラスターまたは Redis OSS (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。

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ー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、アベイラビリティーゾーンのリスト。`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に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ーゾーン ElastiCache を選択します。

以下のプロセスでは、 を使用してシャードを追加して、Valkey または Redis OSS (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を説明します ElastiCache API。

Example - シャードの追加

次の例では、Valkey または Redis OSS (クラスターモードが有効) クラスター にノードグループを追加するため`my-cluster`、オペレーションが完了すると合計 4 つのノードグループがあります。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ー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、 リファレンスModifyReplicationGroupShardConfigurationの ElastiCache API「」を参照してください。

オンラインリシャーディングによるシャードの削除

AWS Management Console、、または を使用して AWS CLI、Valkey または Redis OSS (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ElastiCache API。

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、 を使用してシャードを削除することで、Valkey または Redis OSS (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 AWS Management Console。

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 、Valkey または Redis OSS (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。

Valkey または Redis OSS (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. で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインから、Valkey クラスターまたは Redis OSSクラスター を選択します。
3. シャードを削除する Valkey または Redis OSS (クラスターモードが有効) クラスターのクラスター名の左側にあるボックスではなく、名前を見つけて選択します。

   > **Tip**
   >
   > Valkey または Redis OSS (クラスターモードが有効) クラスターのシャード列の値は 1 以上です。

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、 を使用してシャードを削除することで、Valkey または Redis OSS (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 AWS CLI。

> **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 、Valkey または Redis OSS (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IDsするシャード (ノードグループ) のリスト。

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、Valkey または Redis OSS (クラスターモードが有効) クラスター から 2 つのノードグループを削除するためmy-cluster、オペレーションが完了すると合計 2 つのノードグループがあります。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)

オペレーションを使用して ElastiCache API、Valkey クラスターまたは Redis OSS (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。

以下のプロセスでは、 を使用してシャードを削除することで、Valkey または Redis OSS (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 ElastiCache API。

> **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。

を使用して ElastiCache API、Valkey または Redis OSS (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IDsするシャード (ノードグループ) のリスト。

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、Valkey または Redis OSS (クラスターモードが有効) クラスター から 2 つのノードグループを削除するためmy-cluster、オペレーションが完了すると合計 2 つのノードグループがあります。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

Valkey または Redis OSS (クラスターモードが有効) クラスターのシャードは AWS Management Console、、 AWS CLI、または を使用して再調整できますElastiCache API。

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)
- オンラインのシャード再分散 (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、 を使用してシャードを再調整することで、Valkey または Redis OSS (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 AWS Management Console。

Valkey または Redis OSS (クラスターモードが有効) クラスターのシャード間でキースペースを再調整するには

1. で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインから、Valkey クラスターまたは Redis OSSクラスター を選択します。
3. 再調整する Valkey または Redis OSS (クラスターモードが有効) クラスターの名前の左側のボックスではなく、名前を選択します。

   > **Tip**
   >
   > Valkey または Redis OSS (クラスターモードが有効) クラスターの Shards 列の値は 1 以上です。

4. [再分散] を選択します。
5. 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ます。*Slots in the replication group are uniformly distributed. Nothing to do. (Service: AmazonElastiCache; Status Code: 400; Error Code: InvalidReplicationGroupState; Request ID: 2246cebd-9721-11e7-8d5b-e1b0f086c8cf)*。 その場合は、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、 を使用してシャードを再調整することで、Valkey または Redis OSS (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 AWS CLI。

Example - クラスターのシャードの再分散

次の例では、スロットが可能な限り均等に分散`my-cluster`されるように、Valkey または Redis OSS (クラスターモードが有効) クラスターのスロットのバランスを調整します。`--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 \
    --apply-immediately
```

Windows の場合:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --apply-immediately
```

オンラインのシャード再分散 (ElastiCache API)

オペレーションを使用して ElastiCache API、Valkey クラスターまたは Redis OSS (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。

`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、 を使用してシャードを再調整することで、Valkey または Redis OSS (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 ElastiCache API。

Example - クラスターの再分散

次の例では、スロットが可能な限り均等に分散my-clusterされるように、Valkey または Redis OSS (クラスターモードが有効) クラスターのスロットのバランスを調整します。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

Valkey バージョン 7.2 以降、または Redis OSS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、ダウンタイムを最小限に抑えながら Valkey クラスターまたは Redis OSSクラスターを動的にスケーリングできます。これにより、Valkey または Redis OSSクラスターはスケーリング中で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、「のデータ階層化 ElastiCache」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 — Valkey または Redis OSSクラスターのノードタイプを調整して、より大きなノードタイプを使用するようにすることで、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、クラスターを動的にサイズ変更します。
- [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、オンラインのままにしてリクエストを処理しながら、クラスターを ElastiCache 動的にサイズ変更します。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。
>
> - （ ENIElastic Network Interface) 容量が十分で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、「Valkey と Redis の予約済みメモリの管理 OSS」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

- Valkey または Redis OSSキャッシュクラスターのスケールダウン (コンソール)
- Valkey または Redis OSSキャッシュクラスターのスケールダウン (AWS CLI)
- Valkey または Redis OSS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

- Valkey または Redis OSSキャッシュクラスターのスケールアップ (コンソール)
- Valkey または Redis OSSキャッシュクラスターのスケールアップ (AWS CLI)
- Valkey または Redis OSSキャッシュクラスターのスケールアップ (ElastiCache API)

Valkey または Redis OSS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 Valkey または Redis OSS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、クラスターは最小限のダウンタイムでリクエストを引き続き処理します。

Valkey または Redis OSS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から、Valkey クラスターまたは Redis OSS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. Modify 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. Modify 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。

Valkey または Redis OSSキャッシュクラスターのスケールアップ (AWS CLI)

次の手順では、 を使用して Valkey または Redis OSS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ます AWS CLI。このプロセス中、クラスターは最小限のダウンタイムでリクエストを引き続き処理します。

Valkey または Redis OSS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. 次のパラメータを使用して `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
   ```

   上記のコマンドからの出力は次のようになります (JSON 形式) 。

   ```
   {
        "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、以下を参照してください。list-allowed-node-type- 変更 AWS CLI リファレンスの「」を参照してください。

2. コマンドと次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-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
   ```

上記のコマンドからの出力は次のようになります (JSON 形式) 。

```
{
  "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、以下を参照してください。modify-replication-group リファレンス のAWS CLI「」。

3. を使用した場合は`--apply-immediately`、次のパラメータを使用して 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。

Valkey または Redis OSS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、 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リングします ElastiCache API。このプロセス中に、 はDNSエントリ ElastiCache 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。Valkey 7.2 以降 Redis 5.0.5 OSS 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する間、自動フェイルオーバーが有効になっているクラスターをスケールできます。バージョン Redis OSS4.0.10 以降では、DNSエントリの更新中に、プライマリノードからの以前のバージョンの読み取りと書き込みが一時的に中断されることがあります。

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。

Valkey または Redis OSS 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. 次のパラメータを使用して、 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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」を参照してください。

2. ModifyReplicationGroup ElastiCache API アクションと次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、前のステップの ListAllowedNodeTypeModificationsアクションによって返されるインスタンスタイプの 1 つである必要があります。
   - 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-m3-2xl
    &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、以下を参照してください。ModifyReplicationGroup 「Amazon ElastiCache API リファレンス」の「」を参照してください。

3. を使用した場合はApplyImmediately=true、次のパラメータを使用して ElastiCache APIDescribeReplicationGroups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、以下を参照してください。DescribeReplicationGroups 「Amazon ElastiCache API リファレンス」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

- Valkey または Redis OSSキャッシュクラスターのスケールダウン (コンソール)
- Valkey または Redis OSSキャッシュクラスターのスケールダウン (AWS CLI)
- Valkey または Redis OSSキャッシュクラスターのスケールダウン (ElastiCache API)

Valkey または Redis OSS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 Valkey または Redis OSS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、Valkey または Redis OSSクラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。

Valkey または Redis OSS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から、Valkey クラスターまたは Redis OSS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. Modify 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. Modify 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。

Valkey または Redis OSS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、 を使用して Valkey または Redis OSS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 AWS CLI。このプロセス中、クラスターは最小限のダウンタイムでリクエストを引き続き処理します。

Valkey または Redis OSS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. 次のパラメータを使用して `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
```

上記のコマンドからの出力は次のようになります (JSON 形式) 。

```
{
     "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、以下を参照してください。list-allowed-node-type- 変更 AWS CLI リファレンスの「」。

2. コマンドと次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
```

上記のコマンドからの出力は次のようになります (JSON 形式) 。

```
{
  "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、以下を参照してください。modify-replication-group AWS CLI リファレンス の「」を参照してください。

3. を使用した場合は`--apply-immediately`、次のパラメータを使用して 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。

Valkey または Redis OSSキャッシュクラスターのスケールダウン (ElastiCache API)

以下のプロセスでは、 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なノードタイプにスケーリングします ElastiCache API。このプロセス中、Valkey または Redis OSS クラスターは、ダウンタイムを最小限に抑えながらリクエストを引き続き処理します。

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. 次のパラメータを使用して、 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、以下を参照してください。ListAllowedNodeTypeModifications 「Amazon ElastiCache API リファレンス」の「」を参照してください。

2. `ModifyReplicationGroup` ElastiCache API アクションと次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、前のステップの `ListAllowedNodeTypeModifications`アクションによって返されるインスタンスタイプの 1 つである必要があります。
   - `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-m3-2xl
    &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、以下を参照してください。ModifyReplicationGroup 「Amazon ElastiCache API リファレンス」の「」を参照してください。

JSON for Valkey と Redis の開始方法 OSS

ElastiCache は、ネイティブの JavaScript Object Notation (JSON) 形式をサポートしています。これは、Valkey クラスターと Redis OSSクラスター内で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。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シリアル化解除する必要はありません。

経由で動作するアプリケーションに Valkey および Redis OSSAPIオペレーションを使用するだけでなくJSON、オブジェクト全体を操作することなく、JSONドキュメントの特定部分を効果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。Goessner 形式の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データ型と関連するコマンドが自動的に使用可能になります。API は、 JSONモジュールのバージョン 2 とRDB互換性があるため、既存の JSONベースの Valkey および Redis OSSアプリケーションを簡単に に移行できますElastiCache。サポートされているコマンドの詳細については、「」を参照してくださいサポートされている Valkey コマンドと Redis OSS コマンド。

JSONに関連するメトリクスJsonBasedCmdsと JsonBasedCmdsLatencyは、このデータ型の使用をモニタリング CloudWatch するために に組み込まれています。詳細については、「Valkey と Redis のメトリクスOSS」を参照してください。

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

トピック

- JSON データ型の概要
- サポートされている Valkey コマンドと Redis OSS コマンド

JSON データ型の概要

ElastiCache は、JSONデータ型を操作するために多数の Valkey コマンドと Redis OSS コマンドをサポートしています。以下は、JSONデータ型の概要とサポートされているコマンドの詳細なリストです。

用語

言葉	説明
JSON ドキュメント	JSON キーの値を指します。
JSON 値	JSON ドキュメント全体を表すルートを含む、ドキュメントのサブセットを指します。値は、

言葉	説明
	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 値に相当します。

サポートされているJSON標準

JSON 形式は RFC7159 および ECMA-404 JSON データ交換標準に準拠しています。UTF-8 テキストの Unicode がサポートされています。 JSON

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 4627 RFC では、ルート値として許可されているのはオブジェクトまたは配列のみであることに注意してください。7159 RFC 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、迅速なアクセスと変更のために最適化された形式で内部に保存されます。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。

1 つのJSONドキュメントによるメモリの消費は 64 MB に制限されます。これは、JSON文字列ではなく、メモリ内データ構造のサイズです。`JSON.DEBUG MEMORY` コマンドを使用して、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

- データ型ごとの既存のカテゴリ (@string、@hash など) と同様に、JSONコマンドとデータへのアクセス管理を簡素化するために、新しいカテゴリ @json が追加されました。他の既存の Valkey コマンドや Redis OSS コマンドは、@json カテゴリのメンバーではありません。すべてのJSONコマンドは、キースペースまたはコマンドの制限とアクセス許可を適用します。
- 既存の Valkey カテゴリと Redis OSSACLカテゴリは 5 つあり、新しいJSONコマンドを含めるように更新されています。@read、@write、@fast、@slow、@admin です。次の表は、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.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 コマンド	@read	@write	@fast	@slow	@admin
JSON.NUMMULTBY		y	y		
JSON.OBJKEYS	y		y		
JSON.OBJLEN	y		y		
JSON.RESP	y		y		
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に「ネスト」とされます。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 は、次の 2 種類のパス構文JSONをサポートしています。

- 拡張構文 — 次の表に示すように、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。

記号/式	説明
*	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。
()	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。複数のフィルター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

次の例は、Goessner のサンプルXMLデータに基づいて構築されています。このデータには、フィールドを追加することで変更されています。

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "in-stock": true,
        "sold": false
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99,
        "in-stock": false,
        "sold": false
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95,
      "in-stock": true,
      "sold": false
    }
  }
```

```
}
```

パス	説明
$.store.book[*].author	この店のすべての本の著者です。
$..author	すべての著者です。
$.store.*	店のすべてのメンバー。
$["store"].*	店のすべてのメンバー。
$.store..price	店のすべてのものの価格です。
$..*	JSON 構造のすべての再帰メンバー。
$..book[*]	すべての本です。
$..book[0]	最初の本です。
$..book[-1]	最後の本です。
$..book[0:2]	最初の 2 冊の本です。
$..book[0,1]	最初の 2 冊の本です。
$..book[0:4]	インデックス 0 から 3 までの本です (終了インデックスは含みません)。
$..book[0:4:2]	インデックス 0, 2 の本です。
$..book[?(@.isbn)]	ISBN 番号を持つすべての本。
$..book[?(@.price<10)]	10 ドルより安いすべての本。
'$..book[?(@.price < 10)]'	10 ドルより安いすべての本です。(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。)
'$..book[?(@["price"]< 10)]'	10 ドルより安いすべての本。

パス	説明
'$..book[?(@.["price"]< 10)]'	10 ドルより安いすべての本。
$..book[?(@.price>=10&&@.price<=100)]	10 ドルから 100 ドルの価格帯 (この値を含む) にあるすべての本です。
'$..book[?(@.price>=10 && @.price<=100)]'	10 ドルから 100 ドルの価格帯 (この値を含む) にあるすべての本です。(パスに空白が含まれている場合は、引用符で囲む必要があります。)
$..book[?(@.sold==true||@.in-stock==false)]	すべての本が売れたか、在庫切れです。
'$..book[?(@.sold == true || @.in-stock == false)]'	すべての本が売れたか、在庫切れです。(パスに空白が含まれている場合は、引用符で囲む必要があります。)
'$.store.book[?(@.["category"] == "fiction")]'	フィクションのカテゴリのすべての本です。
'$.store.book[?(@.["category"] != "fiction")]'	ノンフィクションのカテゴリのすべての本です。

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。
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

次のJSON情報メトリクスが用意されています。

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。
json_num_documents	Valkey または Redis のドキュメントの合計数 OSS。

コアメトリクスをクエリするには、次のコマンドを実行します。

```
info json_core_metrics
```

Valkey と Redis ElastiCache とのOSSインタラクション JSON

次のセクションでは、 ElastiCache と Valkey および Redis がJSONデータ型とOSSやり取りする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 (Redis OSS) 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 には、整数と浮動小数点数の個別のデータ型はありません。それらはすべて数値と呼ばれます。

数値表現:

入力時にJSON数値を受信すると、64 ビット符号付き整数または 64 ビットIEEEの倍精度浮動小数点の 2 つの内部バイナリ表現のいずれかに変換されます。元の文字列、およびそのすべての書式は保持されません。したがって、数値がJSONレスポンスの一部として出力されると、内部バイナリ表現から汎用フォーマットルールを使用する印刷可能な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 の範囲外の場合`int64`、自動的に 64 ビットのIEEE倍精度浮動小数点数になります。
- 数値の少なくとも 1 つが浮動小数点の場合、結果は 64 ビットIEEEの倍精度浮動小数点数になります。
- 結果が 64 ビットの IEEE double の範囲を超えると、 コマンドは`OVERFLOW`エラーを返します。

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 Valkey コマンドと Redis OSS 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 Valkey または Redis では、配列オブジェクトを直接OSSフィルタリングします。

のようなデータ`[0,1,2,3,4,5,6]`、 のようなパスクエリ`$[?(@<4)]`、 のようなデータ、 `{"my_key":[0,1,2,3,4,5,6]}` のようなパスクエリ ElastiCache では`$.my_key[?(@<4)]`、Valkey または Redis OSSは両方の状況で [1,2,3] を返します。

配列インデックス作成の動作

ElastiCache Valkey または Redis では、配列の正インデックスと負インデックスの両方OSS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。

顧客の予測可能な動作を確保するために、 ElastiCache Valkey または Redis OSSでは配列インデックスが切り上げられません。そのため、長さが 5 の配列がある場合、インデックスを 5 以上、または -6 以下を呼び出すと、結果が生成されません。

厳密な構文評価

MemoryDB では、JSONパスのサブセットに有効なパスが含まれている場合でも、無効な構文の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 Valkey コマンドと Redis OSS コマンド

ElastiCache では、次の Valkey コマンドと Redis OSS JSON コマンドがサポートされています。

トピック

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR

- JSON.DEBUG
- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE
- JSON.TYPE

JSON.ARRAPPEND

パスの配列値に 1 つ以上の値を追加します。

構文

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

- key (必須) – JSONドキュメントタイプの Valkey または Redis OSSキー。
- path (必須) – JSONパス。
- json (必須) – 配列に追加するJSON値。

戻る

パスが拡張構文の場合:

- 各パスの配列の新しい長さを表す整数の配列。
- 値が配列でない場合、対応する戻り値は null です。

- SYNTAXERR 入力 json 引数の 1 つが有効なJSON文字列でない場合、 エラー。
- 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。
- SYNTAXERR 入力 json 引数の 1 つが有効なJSON文字列でない場合、 エラー。
- パスが存在しない場合は、NONEXISTENT エラーになります。

例

拡張パス構文:

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

制限パス構文:

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

JSON.ARRINDEX

パスの配列でスカラーJSON値の最初の出現を検索します。

- 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 Valkey または Redis OSSキー。
- path (必須) – JSONパス。
- json-scalar (必須) – 検索するスカラー値。JSON スカラーとは、オブジェクトや配列ではない値を指します。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 Valkey または Redis OSS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) – 配列に追加する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 Valkey または Redis OSS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 Valkey または Redis OSS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:

- ポップされたJSON値を表す一括文字列。
- 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 Valkey または Redis OSS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 Valkey または Redis OSS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] – メモリ使用量をJSON値のバイト単位でレポートします。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の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、含まれる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 – コマンドのヘルプメッセージを印刷します。

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JSONの値のメモリサイズ (バイト単位) を表す整数の配列を返します。
  - Valkey または Redis OSS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、メモリサイズ、およびJSON値をバイト単位で返します。
  - Valkey または Redis OSS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JSONの値フィールドの数を表す整数の配列を返します。
  - Valkey または Redis OSS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 値の整数のフィールド数を返します。
  - Valkey または Redis OSS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JSON内のパスの値を削除します。パスがルートの場合、Valkey または Redis からキーを削除するのと同じですOSS。

構文

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

- key (必須) – JSONドキュメントタイプの Valkey または Redis OSS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。
- Valkey または Redis OSSキーが存在しない場合は 0。
- JSON パスが無効または存在しない場合は 0。

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JSONでシリアル化された 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 Valkey または Redis OSSキー。
- INDENT/NEWLINE/SPACE (オプション) — 返されたJSON文字列の形式、つまり「プリティプリント」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。レガシー互換性のために存在でき、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パスJSONsでシリアル化されます。存在しないキーまたはJSONパスに対して 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 Valkey キーまたは Redis OSS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JSONによって位置するシリアル化された 、または (b) キーが存在しない場合、パスがドキュメントに存在しない場合、またはパスが無効 (構文エラー) のいずれかが入力されます。
- 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 Valkey または Redis OSS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。
- OVERFLOW 結果が 64 ビットのIEEE倍数範囲外の場合、 エラー。
- 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。
- OVERFLOW 結果が 64 ビットのIEEE倍数範囲外の場合、 エラー。
- 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 Valkey または Redis OSS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。
- OVERFLOW 結果が 64 ビットのIEEE倍精度浮動小数点数の範囲外の場合、 エラー。
- 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。
- OVERFLOW 結果が 64 ビットのIEEE倍数の範囲外の場合、 エラー。
- 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 Valkey または Redis OSS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 Valkey または Redis OSS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

Valkey または Redis OSS シリアル化プロトコル () 内の指定されたパスJSONの値を返します RESP。値がコンテナの場合、レスポンスはRESP配列またはネストされた配列です。

- JSON null は RESP Null 一括文字列にマッピングされます。
- JSON ブール値は、それぞれの RESP Simple Strings にマッピングされます。
- 整数番号はRESP整数にマッピングされます。
- 64 ビットのIEEE二重浮動小数点数は、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、最初の要素は単純な文字列 [、次に配列の要素です。
- JSON オブジェクトはRESP配列として表され、最初の要素は単純な文字列 { で、キーと値のペアが続き、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 Valkey または Redis OSS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 つのパスの値の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、 コマンドはNONEXISTENTエラーを返します。
- 親要素が存在するがオブジェクトではない場合、 コマンドは を返しますERROR。
- 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、 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、 コマンドはNONEXISTENTエラーを返します。
- 親要素が存在するが配列ではない場合、 コマンドは を返しますERROR。
- 親要素が存在するが、インデックスが範囲外の場合、 コマンドはOUTOFBOUNDARIESエラーを返します。
- 親要素が存在し、インデックスが有効な場合、要素は新しいJSON値に置き換えられます。

パスがオブジェクトまたは配列を呼び出す場合、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 Valkey または Redis OSSキー。
- path (必須) – JSONパス。新しいキーの場合、JSONパスは root「」である必要があります。
- NX (オプション) – パスがルートの場合は、キーが存在しない場合にのみ値を設定します。つまり、新しいドキュメントを挿入します。パスがルートでない場合は、パスが存在しない場合にのみ値を設定します。つまり、ドキュメントに値を挿入します。
- XX (オプション) – パスがルートの場合は、キーが存在する場合にのみ値を設定します。つまり、既存のドキュメントを置き換えます。パスがルートでない場合は、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

パスの文字列にJSON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 Valkey または Redis OSS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。
- SYNTAXERR 入力 json 引数が有効なJSON文字列でない場合、 エラー。
- 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。
- WRONGTYPE 入力 json 引数が有効なJSON文字列でない場合、 エラー。
- 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.STRLEN

パスのJSON文字列値の長さを取得します。

構文

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

- key (必須) – JSONドキュメントタイプの Valkey または Redis OSS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 Valkey または Redis OSS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 Valkey または Redis OSS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 パスが無効または存在しない場合は Null。

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

クラスターやその他の ElastiCache リソースの管理に役立つように、タグの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、目的、所有者、環境など、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、目的や所有者など、さまざまな方法で AWS リソースを分類できます。例えば、各インスタンスの所有者とユーザーグループを追跡するのに役立つ、アカウントの ElastiCache クラスターのタグのセットを定義できます。

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。

タグには意味的な意味はなく ElastiCache 、厳密には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 を使用してタグを操作できます ElastiCache API。

を使用している場合はIAM、 AWS アカウント内のどのユーザーにタグを作成、編集、または削除するためのアクセス許可があるかを制御できます。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

アカウント内に既に存在するほとんどの ElastiCache リソースにタグを付けることができます。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。を使用している場合は AWS Management Console、タグエディタ を使用してリソースにタグを適用できます。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。コンソールは Name タグに従ってリソースを整理できますが、このタグは ElastiCache サービスにとって意味的な意味を持ちません。

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 を使用している場合 AWS SDKは、関連する ElastiCache APIアクションで 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

次の表は、 、、または ElastiCache APIを使用して、タグ付けできる ElastiCache リソースと AWS CLI、作成時にタグ付けできるリソースを示しています AWS SDK。

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
あり	可能

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

> **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 ポリシー内のタグベースのリソースレベルのアクセス許可を、 ElastiCache API作成時のタグ付けをサポートするアクションに適用して、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きめ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では使用できません。

- 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。
- 最大キー長 – UTF-8 で 128 Unicode 文字。
- 最大値の長さ – UTF-8 で 256 個の Unicode 文字。
- ElastiCache はタグ内の任意の文字を許可しますが、他のサービスは制限される場合があります。サービス全体で使用できる文字は、UTF-8 で表される文字、数字、スペース、および + - = . _ : / @ 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 使用のために予約されています。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、「」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](#)。

リソースのタグ付けの例

- タグを使用したサーバーレスキャッシュの作成。この例では、エンジンとして Memcached を使用します。

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine memcached
    --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 valkey \
--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を作成します。この例では、Redis をエンジンとして使用します。

```
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 valkey \
--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。この例では、エンジンとして Memcached を使用します。

```
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。

  現在、スナップショットは Redis でのみ使用できます。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= というタグが付いている場合にのみ、クラスターにAddTagsToResourceアクションを許可しますXYZ。

```
{
    "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アクションを拒否します。

```
{
    "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 のリソースにコスト配分タグを追加すると ElastiCache、請求書の費用をリソースタグ値で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、独自のコスト構造を反映するように AWS 請求書を整理します。これを行うには、サインアップして、タグキー値を含む AWS アカウント請求書を取得します。次に、結合したリソースのコストを見るには、同じタグキー値のリソースに従って請求書情報を整理します。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。

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。その後、合計 ElastiCache コストをリージョン別に分類できます。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。

ElastiCache コスト ElastiCache 配分タグを独自設計のクラスターに追加できます。タグの追加やリスト、変更、削除を行った場合、そのオペレーションは、指定したクラスターにのみ適用されます。

ElastiCache コスト配分タグの特徴

- コスト配分タグは、 CLIおよび APIオペレーションで として指定された ElastiCache リソースに適用されますARN。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 個のタグを含めることができます。

- 値はタグセット内で一意である必要はありません。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、タグにセマンティックな意味を適用しません。タグは厳密に文字列として解釈されます。AWS は ElastiCache リソースにタグを自動的に設定しません。

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 AWS CLI を追加、変更、または削除できます。

コスト配分タグは ElastiCache クラスターに適用されます。タグを付けるクラスターは、 ARN (Amazon リソースネーム) を使用して指定されます。

サンプル arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

トピック

- 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 AWS CLI

を使用して既存の ElastiCache リソースのタグを AWS CLI 一覧表示するには、list-tags-for-resource オペレーション。

次のコードは AWS CLI 、 を使用して、us-west-2 リージョンの Memcached クラスター上のタグ`my-cluster`を一覧表示します。

Linux、macOS、Unix の場合:

```
aws elasticache list-tags-for-resource \
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

Windows の場合:

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

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my-cluster-001内の Valkey または Redis OSSノードのタグを一覧表示します。

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"
      }
   ]
}
```

リソースにタグがない場合、出力は空の になります TagList。

```
{
   "TagList": []
}
```

詳細については、「」を参照してください AWS CLI 。 ElastiCache list-tags-for-resource.

を使用したタグの追加 AWS CLI

を使用して既存の ElastiCache リソースにタグ AWS CLI を追加するには、add-tags-to-resource CLI オペレーション。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、 キーServiceと 値Region、elasticacheおよび us-west-2 をそれぞれ my-clusterリージョン us-west-2 のクラスターmy-cluster-001内のノードに追加します。

Memcached

Linux、macOS、Unix の場合:

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --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 ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

Redis

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、「」を参照してください AWS CLI 。 ElastiCache add-tags-to-resource.

オペレーションを使用して新しいクラスターを作成するときに、 を使用してクラスターにタグ AWS CLI を追加することもできます。create-cache-cluster。 ElastiCache 管理コンソールを使用してクラスターを作成するときにタグを追加することはできません。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。

を使用したタグの変更 AWS CLI

を使用して AWS CLI 、 ElastiCache クラスターのタグを変更できます。

タグを変更するには:

- 使用アイテム add-tags-to-resource 新しいタグと値を追加するか、既存のタグに関連付けられた値を変更します。
- 使用アイテム remove-tags-from-resource リソースから指定され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。

を使用したタグの削除 AWS CLI

を使用して既存の ElastiCache (Memcached) クラスターからタグ AWS CLI を削除するには、remove-tags-from-resource オペレーション。

Memcached の場合、次のコードは AWS CLI を使用して、us-west-2 リージョンmy-clusterのクラスターmy-cluster-001内のノードRegionから キーServiceと を持つタグを削除します。

Linux、macOS、Unix の場合:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --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 ^
 --tag-keys PM Service
```

Redis の場合、次のコードは AWS CLI を使用して、us-west-2 リージョンmy-clusterのクラスターmy-cluster-001内のノードRegionからキーServiceと を含むタグを削除します。

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、「」を参照してください AWS CLI 。 ElastiCache remove-tags-from-resource.

を使用したコスト配分タグの管理 ElastiCache API

を使用して ElastiCache API、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、Memcached クラスター ElastiCache の に適用されます。タグを付けるクラスターは、 ARN (Amazon リソースネーム) を使用して指定されます。

サンプル arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

トピック

を使用したタグの一覧表示 ElastiCache API

を使用して ElastiCache API既存のリソースのタグを一覧表示するには、ListTagsForResource オペレーション。

Memcached の場合、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のタグ`my-cluster`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
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Redis の場合、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のタグ`my-cluster-001`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>
```

を使用したタグの追加 ElastiCache API

を使用して ElastiCache API既存の ElastiCache クラスターにタグを追加するには、AddTagsToResource オペレーション。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、 ElastiCache API を使用して、 キーServiceと をそれぞれ値elasticacheと Region で追加しますus-west-2。Memcached の場合、これはリソース に適用されますmy-cluster。Redis の場合、これは us-west-2 リージョンmy-cluster-001のリソースに適用されます。

Memcached

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
   &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>
```

Redis

```
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、以下を参照してください。AddTagsToResource 「Amazon ElastiCache API リファレンス」の「」を参照してください。

を使用したタグの変更 ElastiCache API

を使用して ElastiCache API、 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:

- 使用アイテム AddTagsToResource オペレーションは、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。
- 使用アイテム RemoveTagsFromResource 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。

使用アイテム RemoveTagsFromResource リソースからタグを削除します。

を使用したタグの削除 ElastiCache API

を使用して ElastiCache API既存の ElastiCache (Memcached) クラスターからタグを削除するには、RemoveTagsFromResource オペレーション。

次のコードは、 ElastiCache APIを使用して、 `my-cluster` リージョン us-west-2 のクラスター`my-cluster-001`内のノード`Region`から キー`Service`と を持つタグを削除します。

```
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、Amazon ElastiCache Well-Architected レンズについて説明します。Amazon Well-Architected レンズは、適切に設計された 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コレクションです。

- ElastiCache レンズは AWS Well-Architected フレームワーク に追加されます。
- 各柱には、アーキテクチャに関するディスカッションを開始するのに役立つ一連の質問があります ElastiCache。
  - 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 – 要件に対してまとめて提供するコード、設定、 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 - ビジネス価値を提供する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。

> **Note**
>
> このガイドは、 ElastiCache サーバーレスキャッシュと新しい Valkey エンジンに関する情報を含めるように更新されていません。

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

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理解し、対応しますか？

質問レベルの紹介: ElastiCache クラスターを操作すると、特定のイベントが発生したときに通知やアラートをオプションで受け取ることができます。デフォルトで ElastiCache、 は、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、スケジュールされたメンテナンスなど、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ElastiCache で (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) describe-events コマンドと を使用して、 によって生成されたイベントを確認しますElastiCache API。Amazon Simple Notification Service (Amazon)

を使用して重要なクラスターイベントの通知を送信する ElastiCache ように を設定しますSNS。クラスターSNSで Amazon を使用すると、 ElastiCache イベントに対してプログラムで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、スナップショットの失敗) 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（ バルキーまたは Redis OSSのみ ）
- [リソース]:
  - ElastiCache Amazon SNS通知の管理
  - イベント通知と Amazon SNS

- 〔ベスト〕 イベントへの対応を自動化するには、 SNS や Lambda Functions 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。Amazon CloudWatch メトリクスを使用してクラスターをモニタリングする必要があります。

  〔リソース]: AWS Lambda、Amazon Route 53、Amazon を使用して、Lambda と を使用するユースケースについて、リードレプリカエンドポイントをモニタリング ElastiCache (Redis OSS) (クラスターモードが無効) SNS しますSNS。

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しますか？

質問レベルの紹介: ElastiCache クラスターのサイズを適正化することは、基盤となるワークロードタイプが変更されるたびに評価する必要があるバランスの取れた行為です。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、リソースのオーバー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。リソースの過剰または過少使用率を修正するには、 ElastiCache を 2 つのディメンションにスケールできます。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〕 CPUおよびプライマリノードでのネットワークの過剰利用は、リードオペレーションをオフロードしてレプリカノードに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、クラスターモードが無効の場合は ElastiCache リーダーエンドポイントに接続するか、クラスターモードが有効の場合は READONLY コマンドを使用して、Valkey または Redis OSSクライアントライブラリで設定できます。

  [リソース]:

  - での接続エンドポイントの検索 ElastiCache
  - クラスターの適切なサイズ設定
  - READONLY コマンド
- 〔必須] 、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率をモニタリングします。 CPU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、スケーリングの決定およびスケーリングオペレーションのタイプを通知する必要があります。 ElastiCache (Redis OSS) クラスターモードが無効の場合、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リング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 を使用した ElastiCache (Redis OSS) によるベストプラクティスのモニタリング CloudWatch
  - スケーリング ElastiCache (Redis OSS) クラスター
  - Memcached クラスターの ElastiCache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。長期的な傾向を検出するには、メトリクスを使用して CloudWatch、より長い期間をスキャンします。長期間の CloudWatch メトリクスを観察すること

から学んだことは、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。 CloudWatch データポイントとメトリクスは最大 455 日間使用できます。

[リソース]:

- メトリクスによる CloudWatchモニタリング ElastiCache (Redis OSS)
- CloudWatch メトリクスによる Memcached のモニタリング
- Amazon を使用した ElastiCache (Redis OSS) によるベストプラクティスのモニタリング CloudWatch

- 〔最適〕 ElastiCache リソースが で作成CloudFormation されている場合は、運用上の一貫性を維持し、管理されていない設定の変更やスタックドリフトを回避するために、テンプレートを使用して CloudFormation変更を実行することをお勧めします。

  [リソース]:

  - ElastiCache のリソースタイプリファレンス CloudFormation

- 〔ベスト〕 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、 でしきい値を定義 CloudWatch してアラームを設定します。CloudWatch Events and 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 アラームの使用
  - Amazon CloudWatch イベントとは
  - Amazon Simple Notification Service AWS Lambda での 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、クラスターを維持するにはどうすればよいですか up-to-date？

質問レベルの紹介: 大規模に運用する場合は、すべての ElastiCacheリソースを特定および特定できることが重要です。新しいアプリケーション機能をロールアウトするときは、開発、テスト、本番稼働のすべての ElastiCache 環境タイプにわたってクラスターバージョンの対称性を作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。ネイティブ ElastiCache 機能を活用することで、エンジニアリングチームは ElastiCache 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。

- 〔ベスト〕 利用可能な最新バージョンのエンジンで を実行し、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。 は、クラスターの指定されたメンテナンスウィンドウ中に基盤となるインフラストラクチャ ElastiCache 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 でのセルフサービスの更新 ElastiCache
  - Amazon ElastiCache Managed Maintenance and Service Updates ヘルプページ
- 〔ベスト〕 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
  - ElastiCache のリソースタイプリファレンス CloudFormation
  - 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しますか？

質問レベルの紹介: 大規模に運用する場合は、クライアントが ElastiCache クラスターと接続してアプリケーションの運用面 (レスポンスタイムなど) を管理する方法を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、Valkey

レプリケーションと Redis OSSレプリケーションの非同期性のために、キーを書き込んだ直後にキーを読み取る機能が失われることに注意してください。WAIT コマンドを活用して、実際のデータの安全性を向上させ、レプリカがクライアントに応答する前に書き込みを強制的に承認するように、全体的なパフォーマンスコストで行うことができます。リードオペレーションにレプリカノードを使用することは、クラスターモードの ElastiCache リーダーエンドポイントを無効にして ElastiCache (Redis OSS) 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 (Redis OSS) READONLY コマンドを使用します。 (ElastiCache Redis OSS) クライアントライブラリの多くでは、 ElastiCache (Redis OSS) READONLYはデフォルトで、または設定を介して実装されます。

[リソース]:

- での接続エンドポイントの検索 ElastiCache
- 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CPUの両方で時間がかかり、プーリングにより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。 ElastiCache (Redis OSS) クライアントライブラリ (サポートされている場合) を介して接続プーリングを実装し、アプリケーション環境で使用できるフレームワークを使用して、またはゼロから構築できます。

- [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、接続の開閉に必要なレイテンシーとCPU使用率が軽減され、クラスターで TLS が有効になっている場合はTLSハンドシェイクが実行されます。

  〔リソース]: 可用性を高めるために ElastiCache (Redis OSS) を設定する

- [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ラウンドトリップ時間 (RTT) を短縮でき、クライアントが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: ワークロードの ElastiCache コンポーネントをデプロイする方法

質問レベルの紹介: ElastiCache 環境は、 AWS コンソールを介して手動でデプロイすることも、APIs、CLIツールキットなどを介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によって分離することも、コスト最適化のために組み合わせることもできます。

質問レベルの利点: ElastiCache 環境に最適なデプロイメカニズムを選択すると、時間の経過とともにオペレーションエクセレンスが向上します。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 ElastiCache 環境を用意するか、複数のワークロードを単一のクラスターに結合するか、その組み合わせを選択できます。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〕 で使用できるデプロイオプションを理解し ElastiCache、可能な限りこれらの手順を自動化します。自動化の可能な方法には CloudFormation、、 AWS CLI/SDK、および が含まれます APIs。

  [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。ワークロードごとに、ElastiCache リソースへのアクセス、サイジング、スケーリング、管理を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) 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: Operational Excellence には、潜在的な障害の原因を特定して、障害の発生源を特定するための定期的な「死前の」演習を実施することによる障害の予測が含まれます。 は、テスト目的で、シミュレートされたノード障害イベントAPIを許可するフェイルオーバー ElastiCache を提供します。

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: Valkey または Redis OSS 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しますか？

質問レベルの紹介: Operational Excellence では、サービスレベルとエンジンレベルの情報の両方を調査して、クラスターのヘルスとステータスを分析する機能が必要です。 ElastiCache は、Amazon CloudWatch と Amazon Kinesis Data Firehose の両方に Valkey または Redis OSS エンジンログを出力できます。

質問レベルの利点: ElastiCache クラスターで Valkey または Redis OSS エンジンログを有効にすると、クラスターのヘルスとパフォーマンスに影響を与えるイベントに関するインサイトが得られます。Valkey または Redis OSS エンジンログは、 ElastiCache イベントメカニズムを介して利用できないエンジンから直接データを提供します。 ElastiCache イベント (前述の OE-1 を参照) とエンジンログの両方を注意深く観察することで、ElastiCache サービスの観点からもエンジンの観点からも、トラブルシューティング時にイベントの順序を決定できます。

- 〔必須〕 Redis OSS エンジンログ記録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 Redis OSS) 6.2 以降で使用できます。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 または Amazon Kinesis Data Firehose が Redis OSS エンジンログの適切なターゲットであるかどうかを判断します。
  - ログを保持するには、 CloudWatch または Kinesis Data Firehose のいずれかで適切なターゲットログを選択します。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
  - ログ記録先: Amazon CloudWatch Logs
  - Amazon CloudWatch Logs の概要: Amazon CloudWatch Logs とは
  - Amazon Kinesis Data Firehose の紹介: Amazon Kinesis Data Firehose とは
- 〔最適〕 Amazon CloudWatch Logs を使用する場合は、Amazon CloudWatch Logs Insights を活用して、重要な情報を Valkey または Redis OSS エンジンログにクエリすることを検討してください。

  例えば、次のようなWARNING LogLevel 「」のイベントを返す Valkey または Redis OSS エンジンログを含む CloudWatch Log 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 レンズセキュリティ柱

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 データへの認可されたアクセスを制御するために、どのような手順を実行していますか？

質問レベルの紹介: すべての ElastiCacheクラスターは、、サーバーレス関数 (AWS Lambda) VPC、またはコンテナ (Amazon Elastic Container Service) の Amazon Elastic Compute Cloud インスタンスからアクセスできるように設計されています。最も遭遇するシナリオは、同じ Amazon Virtual Private Cloud (Amazon Virtual Private Cloud) 内の 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前に、Amazon EC2インスタンスがクラスターにアクセスすることを承認する必要があります。で実行されている ElastiCache クラスターにアクセスするにはVPC、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。

質問レベルの利点: 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を介して制御されます。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、送受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。の場合 ElastiCache、クラスターを起動するときには、セキュリティグループの関連付けが必要です。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、 VPCのプライベートネットワークを介してのみ からアクセスできるように、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、 への進入パスはありません ElastiCache。これを有効にするには、ソース IP アドレス/範囲を指定するセキュリティグループにインバウンドルールを設定し、トラフィック

と ElastiCache クラスターのポート (ElastiCache (Redis OSS) のデフォルトポート 6379) TCPを入力します。 (VPC0.0.0.0/0) 内のすべてのリソースと同様に、非常に広範な進入ソースセット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 EC2インスタンスで実行されている Valkey または Redis OSS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を定義する際には、できるだけきめ細かく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 ポリシーは、 ElastiCache データへのアクセスを許可する AWS Lambda 関数に割り当てることができます。この機能を有効にするには、 アクセスAWSLambdaVPCAccessExecutionRole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 ElastiCache で Amazon にアクセスするように Lambda 関数を設定する VPC: チュートリアル: Amazon ElastiCache で Amazon にアクセスするように Lambda 関数を設定する VPC

SEC 2: アプリケーションには、ネットワークベースのコントロール以上に対する ElastiCache追加の認可が必要ですか？

質問レベルの紹介: 個々のクライアントレベルで ElastiCache (Redis OSS) クラスターへのアクセスを制限または制御する必要がある場合は、 ElastiCache (Redis OSS) AUTH コマンドを使用して認証することをお勧めします。ElastiCache (Redis OSS) 認証トークンは、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、クライアントがコマンドとアクセスキーを実行できるようにする前に ElastiCache (Redis OSS) がパスワードを要求できるようにし、データプレーンのセキュリティを向上させます。

質問レベルの利点: データを安全に維持するために、 ElastiCache (Redis OSS) は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。これには、許可されたコマンドを実行する ElastiCache 前にAUTH、クライアントが に接続するためにロールベースのアクセス制御 (RBAC) 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。

- 〔ベスト] ElastiCache (Redis OSS) 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認可のコントロールを定義します。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。を使用するにはRBAC、クラスターの作成時に選択し、転送中の暗号化を有効にする必要があります。が を活用TLSできるように、 をサポートする Valkey または Redis OSSクライアントを使用していることを確認しますRBAC。

  [リソース]:

  - (ElastiCache Redis OSS) のレプリケーショングループRBACへの適用
  - 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 (Redis OSS) バージョン
- 〔ベスト] 6.x より前の ElastiCache (Redis OSS) バージョンでは、強力なトークン/パスワードを設定し、 ElastiCache (Redis OSS) の厳格なパスワードポリシーを維持するだけでなくAUTH、パスワード/トークンをローテーションすることをお勧めします。 は、いつでも最大 2 つの (2) 認証トークンを管理 ElastiCache 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 (Redis OSS) クラスターのAUTHトークンを変更する

SEC 3: コマンドが誤って実行され、データの損失や障害が発生するリスクはありますか？

質問レベルの紹介: 誤って実行された場合や悪意のある攻撃者によって実行された場合、オペレーションに悪影響を及ぼす可能性のある Valkey または Redis OSS 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、デベロッパーは 開発環境で定期的に FLUSHALL コマンドを呼び出し、ミスにより誤って本番システムでこのコマンドを呼び出しようとし、誤ってデータが失われる可能性があります。

質問レベルの利点: ElastiCache (Redis OSS) 5.0.3 以降、ワークロードを混乱させる可能性のある特定のコマンドの名前を変更することができます。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 (Redis OSS) バージョン 5.0.3 (廃止、バージョン 5.0.6 を使用)

- Redis 5.0.3 OSS パラメータの変更
- Redis OSS セキュリティ

SEC 4: で保管中のデータ暗号化を確保する方法 ElastiCache

質問レベルの紹介: ElastiCache (Redis OSS) はメモリ内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 (ストレージ上で) 保持される可能性のある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。

質問レベルの利点: ElastiCache (Redis OSS) は、保管時の暗号化オプションを提供し、データセキュリティを向上させます。

- 〔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) が作成された場合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、 ElastiCache は保管時の暗号化で使用されるキーを提供および管理します。

  [リソース]:

  - 保存時の暗号化の制限
  - 保管時の暗号化を有効にする
- 〔ベスト] メモリ内の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のために、 AWS Key Management Service （ KMS) を使用してカスタマーマスターキー () を自己管理できますCMK。と ElastiCache の統合により AWS Key Management Service、 ElastiCache (Redis OSS) クラスターの保存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]:

  - からのカスタマーマネージドキーの使用 AWS Key Management Service
  - AWS キー管理サービス
  - AWS KMS の概念

SEC 5: で転送中のデータを暗号化するにはどうすればよいですか ElastiCache？

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。これは、分散システムのコンポーネント内、および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を表します。 ElastiCache （ Redis OSS) 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自体間で転送中のデータを暗号化できるようにすることで、この要件をサポートしています。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。

質問レベルの利点: Amazon 転送 ElastiCache 中の暗号化は、ある場所から別の場所への転送時に、最も脆弱なポイントでデータのセキュリティを高めることができるオプションの機能です。

- 〔必須] 転送中の暗号化は、作成時に ElastiCache (Redis OSS) 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。影響を理解するには、 を有効にする前と後にワークロードをベンチマークすることをお勧めします encryption-in-transit。

  [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制限するにはどうすればよいですか？

質問レベルの紹介: IAM ElastiCache (Redis OSS) クラスターの作成、変更、削除をより厳密に制御できるように、 ElastiCache (Redis OSS) のきめ細かなアクセスコントロールを でポリシー化し、ARN有効にします。

質問レベルの利点: レプリケーショングループ、ノードなどの Amazon ElastiCache リソースの管理は、IAMポリシーに基づいて特定のアクセス許可を持つ AWS アカウントに制限され、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ることで Amazon ElastiCache リソースへのアクセスを管理し、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ます。

  [リソース]:

  - リソースへのアクセス ElastiCache許可の管理の概要

- Amazon 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

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ますか？

質問レベルの紹介: ElastiCacheRBACを有効にしてデプロイすると、 は CloudWatch メトリクスをエクスポートしてセキュリティイベントをユーザーに通知します。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、アクセスキー、またはコマンドの実行の試みの失敗を特定するのに役立ちます。

さらに、 AWS 製品およびサービスリソースは、デプロイを自動化し、後のレビュー/監査のためにすべてのアクションと変更を記録することで、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] RBAC認証と認可の失敗に関連する公開されたCloudWatch メトリクスに精通します。
  - AuthenticationFailures = Valkey または Redis への認証に失敗した OSS
  - KeyAuthorizationFailures = ユーザーがアクセス許可なしでキーにアクセスできなかった
  - CommandAuthorizationFailures = ユーザーがアクセス許可なしでコマンドを実行しようとしても失敗

  [リソース]:
  - Valkey または Redis のメトリクス OSS
- [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 アラームの使用
- 〔ベスト] Valkey または Redis OSS ACL LOG コマンドを使用して詳細を収集します。

  [リソース]:
  - ACL LOG
- 〔最良] ElastiCache デプロイとイベントのモニタリング、ログ記録、分析に関連する AWS 製品およびサービスの機能に精通します。

  [リソース]:

- を使用した Amazon ElastiCache API コールのログ記録 AWS CloudTrail
- elasticache-redis-cluster-automatic-backup-check
- CloudWatch メトリクスの使用のモニタリング

Amazon ElastiCache Well-Architected レンズの信頼性の柱

信頼性の柱は、意図した機能を実行するワークロードと、需要を満たすための障害から迅速に復旧する方法に焦点を当てています。主なトピックには、分散システム設計、復旧計画、要件の変化への適応などがあります。

トピック

- REL 1: 高可用性 (HA) アーキテクチャのデプロイをどのようにサポートしていますか？
- REL 2: で目標復旧ポイント (RPOs) をどのように達成していますかElastiCache？
- REL 3: ディザスタリカバリ (DR) 要件をどのようにサポートしますか？
- 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ですか？
- REL 5: ElastiCache コンポーネントはスケールするように設計されていますか？

REL 1: 高可用性 (HA) アーキテクチャのデプロイをどのようにサポートしていますか？

質問レベルの紹介: Amazon の高可用性アーキテクチャを理解すること ElastiCache で、可用性イベント中に回復力のある状態で運用できます。

質問レベルの利点: ElastiCache クラスターを設計して障害に耐性を持たせることで、 ElastiCache デプロイの可用性が向上します。

- 〔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: ElastiCache (Memcached) と ElastiCache (Redis OSS)

  1. ElastiCache (Memcached) はレプリケーションメカニズムを提供しておらず、主に一時的なワークロードに使用されます。
  2. ElastiCache (Redis OSS) 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の場合、シャードが 1 つだけ必要な小規模なスループット要件ワークロードでも、シャードごとに少なくとも 2 つのレプリカを持つクラスターモードで ElastiCache (Redis OSS) 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. シャードワークロードの場合、Valkey または Redis クラスタープロトコルでは、クォーラムを達成するためにほとんどのプライマリノードを使用可能にする必要があるため、フェイルオーバーイベント中に最低 3 OSS つのシャードを使用すると復旧が速くなります。

  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。

     ElastiCache (Redis OSS) は、これらのノードで最適化されたパフォーマンスを追加しました。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対応するために、モニタリングと適切なサイズ: 負荷が高いと、 ElastiCache (Redis OSS) エンジンが応答しなくなる可能性があり、可用性に影響します。`BytesUsedForCache` および `DatabaseMemoryUsagePercentage` はメモリ使用量の優れた指標ですが、 `ReplicationLag`は書き込みレートに基づくレプリケーションの状態の指標です。これらのメトリクスを使用してクラススケーリングをトリガーできます。

  6. 本番稼働フェイルオーバーイベント API の前にフェイルオーバーでテストすることで、クライアント側の回復性を確保します。

  [リソース]:

  - 可用性を高めるために ElastiCache (Redis OSS) を設定する
  - レプリケーショングループを使用する高可用性

REL 2: で目標復旧ポイント (RPOs) をどのように達成していますかElastiCache？

質問レベルの紹介: ワークロードを理解しRPO、 ElastiCache バックアップとリカバリ戦略の決定を知らせます。

質問レベルの利点: インプレースRPO戦略を持つことで、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。バックアップポリシーと復元ポリシーを設計すると、ElastiCache データのリカバリポイント目標 (RPO) を満たすのに役立ちます。 ElastiCache （ Redis OSS) は、設定可能な保持ポリシーとともに、Amazon S3 に保存されるスナップショット機能を提供します。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RPOの 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 Backup と Restore の機能を確認します。
- [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] のスケジュールされたバックアップ機能を活用して ElastiCache 、定義されたウィンドウ中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 OSS
  - ElastiCache (Redis OSS) 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
  - バックアップと復元 ElastiCache (Redis OSS) クラスター

REL 3: ディザスタリカバリ (DR) 要件をどのようにサポートしますか？

質問レベルの紹介: 災害対策は、ワークロード計画の重要な側面です。 ElastiCache （ Redis OSS) には、ワークロードの耐障害性要件に基づいてディザスタリカバリを実装するためのいくつかのオプションがあります。Amazon ElastiCache Global Datastore を使用すると、1 つのリージョンの ElastiCache (Redis OSS) クラスターに書き込むことができ、他の 2 つのリージョン間のレプリカクラスターからデータを読み取ることができるため、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 戦略を開発して文書化します。 ElastiCache は、一部のユースケースは完全に一時的であり、DR 戦略を必要としないのに対し、他のユースケースはスペクトルの反対側にあり、非常に堅牢な DR 戦略を必要とするという点で一意です。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 アーキテクチャでクラスターモードを有効にした状態でデプロイし、少なくとも 3 つAZsを使用可能にしてください。
  - 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 (Redis OSS) 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ですか？

質問レベルの紹介: 自動フェイルオーバーでマルチ AZ を有効にするのが ElastiCache ベストプラクティスです。場合によっては、 ElastiCache (Redis OSS) はサービスオペレーションの一部として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万一の場合が含まれます。正常なフェイルオーバーは、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。

質問レベルの利点: 特定の ElastiCache (Redis OSS) クライアントライブラリと併せてフェ ElastiCache イルオーバーのベストプラクティスに従うことで、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、 ElastiCache (Redis OSS) クライアントライブラリの設定によって最も頻繁に行われます。これにより、頻度と更新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 (Redis OSS) でのダウンタイムの最小化
  - ElastiCache (Redis OSS) 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ます。
- [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。Valkey と Redis レOSSプリケーションの非同期性、および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の遅延についてレポートできる CloudWatch メトリクスを確認して理解します。より高度なデータ安全性を必要とするユースケースでは、 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を強制的に承認させます。

  [リソース]:

  - Valkey または Redis のメトリクス OSS
  - Amazon を使用した ElastiCache (Redis OSS) によるベストプラクティスのモニタリング CloudWatch
- 〔最良] テストフェイルオーバー を使用して ElastiCache、フェイルオーバー中にアプリケーションの応答性を定期的に検証しますAPI。

[リソース]:

- Amazon 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 ElastiCache (Redis OSS)
- 自動フェイルオーバーのテスト

REL 5: ElastiCache コンポーネントはスケールするように設計されていますか？

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、コンポーネントは ElastiCache変化するワークロード要件に合わせて時間の経過とともに調整できます。ElastiCache では、イン/アウト (水平) とアップ/ダウン (垂直) の 4 ウェイスケーリングが用意されています。

質問レベルの利点: ElastiCache デプロイのベストプラクティスに従うことで、スケーリングの柔軟性を最大限に高めるだけでなく、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 の原則を満たします。

- [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。
  - その他の READIOPS: レプリカを追加する
  - 詳細についてはWRITEOPS、シャードを追加する (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] クラスターモードを有効にして ElastiCache コンポーネントをデプロイします。このとき、ノードの数が少なく、ノードの数が少なく、ノードの数が少なくなります。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リード容量を増やします。 ElastiCache は、クラスターモードが無効になっている場合、および垂直スケーリングで最大 5 つのリードレプリカをサポートしています。
- [リソース]:
  - スケーリング ElastiCache (Redis OSS) クラスター

- オンラインスケールアップ
- Memcached クラスターElastiCache 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の選択、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しますか？
- PE 2: ElastiCache クラスターノード全体にどのように作業を分散していますか？
- 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: のデータをどのようにモデル化して操作しますか ElastiCache？
- PE 7: Amazon ElastiCache クラスターで実行が遅いコマンドを記録するにはどうすればよいですか？
- PE8: Auto Scaling はElastiCache クラスターのパフォーマンスを向上させる上でどのように役立ちますか？

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しますか？

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。
  - これらのテストの実行中に CloudWatch メトリクスをモニタリングして、使用可能なメトリクスを理解し、パフォーマンスベースラインを確立します。
- 〔ベスト] ElastiCache (Redis OSS) ワークロードの場合、 などの計算コストの高いコマンドの名前を変更してKEYS、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機能を制限します。
  - ElastiCache エンジン 6.x を実行している (Redis OSS) ワークロードでは、ロールベースのアクセスコントロールを活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 でユーザーとユーザーグループを作成しCLI、ユーザーグループを ElastiCache (Redis OSS) クラスターに関連付けることで制御できます。Redis OSS6 RBACでは、 を有効にすると、「-@dangerous」を使用できSORT、そのユーザーの KEYS、MONITOR、 などの高価なコマンドは許可されません。
  - エンジンバージョン 5.x では、 ElastiCache (Redis OSS) クラスターrename-commandsパラメータグループの 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 (Redis OSS) ワークロードについては、スローログを分析してクエリの詳細を確認してください。例えば、`valkey-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 (Redis OSS) データ構造を使用して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 (Redis OSS) ワークロードの場合、 `redis-benchmark`は、クライアントの数やデータのサイズなどのユーザー定義の入力を使用して、さまざまなコマンドのパフォーマンスをテストするための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 メトリクスを使用した使用のモニタリング
  - Amazon CloudWatch アラームの使用
  - Valkey および Redis OSS固有のパラメータ

- SLOWLOG
- ベンチマーク

PE 2: ElastiCache クラスターノード全体にどのように作業を分散していますか？

質問レベルの紹介: アプリケーションが 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] クライアントに適切な ElastiCache エンドポイントに接続してもらいます。
  - ElastiCache (Redis OSS) は、使用中のクラスターモードに基づいて異なるエンドポイントを実装します。クラスターモードが有効になっている場合、 ElastiCache は設定エンドポイントを提供します。クラスターモードが無効の場合、 ElastiCache は、通常書き込みに使用されるプライマリエンドポイントと、レプリカ間で読み取りのバランスをとるための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リングが容易になります。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 Memcached クラスターの場合、 は Auto Discovery を有効にする設定エンドポイント ElastiCache 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。
- 〔より良い] （ ElastiCache Redis OSS) クラスターモードを有効にしてスケーラビリティを向上させます。
  - ElastiCache (Redis OSS) (クラスターモードが有効) クラスターは、オンラインスケーリングオペレーション (アウト/インおよびアップ/ダウン) をサポートし、シャード間でデータを動的に分散するのに役立ちます。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トポロジーの変化に確実に適応できるようになります。
  - (Redis OSS) ElastiCache （ クラスターモードが有効) クラスターで使用可能なシャード間でハッシュスロットを移動することで、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 Valkey または Redis OSS データ構造の影響を考慮します。これらのデータ構造は、単一のノードに存在する単一のキーに保存されます。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、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 (Redis OSS) ワークロードの場合、LFUmax-memory ポリシー`valkey-cli --hotkeys`が設定されている場合、 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。このアプローチでは、クライアントが複数のプライマリノードに書き込む必要があり (Valkey または Redis OSSノード自体はこの機能を提供しません)、元のキー名に加えて、読み取るキー名のリストを保持する必要があります。
- ElastiCache Valkey 7.2 以降および Redis OSSバージョン 6 以降では、サーバー支援のクライアント側のキャッシュ がサポートされています。これにより、アプリケーションは へのネットワーク呼び出しを元に戻す前に、キーの変更を待つことができますElastiCache。

- [リソース]:
  - 可用性を高めるためOSSに Valkey と Redis ElastiCache で を設定する
  - での接続エンドポイントの検索 ElastiCache
  - 負荷分散のベストプラクティス
  - Valkey または Redis のオンライン再シャーディング OSS (クラスターモードが有効)
  - Valkey と Redis でのクライアント側のキャッシュ OSS

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 でよく発生するワークロードであり ElastiCache 、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]:

- Valkey と Redis のメトリクス OSS

- 〔必須] レイテンシーとCPU使用率の値と組み合わせてアプリケーションキャッシュのパフォーマンスを測定および収集して、 time-to-liveまたは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把握します。 は、データ構造ごとに集約されたレイテンシーの CloudWatch メトリクスのセット ElastiCache を提供します。これらのレイテンシーメトリクスは、 ElastiCache (Redis OSS) INFO コマンドのコマンド統計を使用して計算され、ネットワークと I/O 時間は含まれません。これは、 ElastiCache (Redis OSS) がオペレーションを処理するために消費した時間のみです。

  [リソース]:

  - Valkey と Redis のメトリクス OSS
  - Amazon を使用した ElastiCache (Redis OSS) によるベストプラクティスのモニタリング CloudWatch

- [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、キャッシュミスの数が多いためです。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]:

  - Memcached のキャッシュ戦略
  - キャッシュのベストプラクティス
  - Amazon ElastiCache ホワイトペーパーを使用した大規模なパフォーマンス

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 (Redis OSS) と ElastiCache (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プロアクティブに管理します。
  - 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 CloudWatch を使用して Amazon の接続動作をモニタリングします`NewConnections`。
  - 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]:
  - ElastiCache (Redis OSS) - グローバルデータストア
  - Memcached 固有のパラメータ
  - ElastiCache (Redis OSS) 5.0.3 は I/O 処理を強化してパフォーマンスを向上させます
  - Valkey と Redis のメトリクス OSS
  - 可用性を高めるために ElastiCache (Redis OSS) を設定する

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の制限に近づいている場合に期待される動作があります。 ElastiCache （ Redis OSS) には、これらの状況を処理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、 `maxmemory-policy` ElastiCache (Redis OSS) クラスターパラメータグループの を使用して管理できます。
  - デフォルトのポリシーは、設定された有効期限 (TTL 値) でキーを削除することでメモリを`volatile-lru`解放します。最小使用頻度 (LFU) ポリシーと最近使用頻度 (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、各ノードの立ち退きを制御するデフォルトのLRUポリシーがあります。Amazon ElastiCache クラスターのエビクションの数は、Amazon 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 CloudWatch。
- [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 (Redis OSS) ワークロードの場合、クラスターからキーを明示的に削除すると、`UNLINK`のようなものになります`DEL`。指定されたキーを削除します。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 (Redis OSS) 6.x ワークロードの場合、 `DEL` コマンドの動作は パラメータグループで `lazyfree-lazy-user-del` パラメータを使用して変更できます。
- [リソース]:
  - パラメータグループを使用したエンジン ElastiCache パラメータの設定
  - UNLINK
  - を使用したクラウド財務管理 AWS

PE 6: のデータをどのようにモデル化して操作しますか ElastiCache？

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 (Redis OSS) 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。

質問レベルの利点: のデータモデリングElastiCache には、アプリケーションのユースケース、データ型、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、各 ElastiCache (Redis OSS) データ型とコマンドには、明確に文書化された独自のパフォーマンス署名があります。

- [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
- [Best] ElastiCache (Redis OSS) コマンドには、Big O 表記で定義される時間の複雑さがあります。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 (Redis OSS) のシングルスレッド設計により、長時間の複雑なオペレーションが多いと、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] クラスター内のデータモデルをGUI可視化APIsするために使用します。

  [リソース]:

  - Redis OSS Commander
  - Redis OSSブラウザ
  - Redsmin

PE 7: Amazon ElastiCache クラスターで実行が遅いコマンドを記録するにはどうすればよいですか？

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。コマンドの実行にかかる時間を理解することで、どのコマンドがパフォーマンスの低下につながるか、およびエンジンが最適に動作することをブロックするコマンドを特定できます。 ElastiCache （ Redis OSS) 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] エンジンバージョン 6.0 以降を実行している Amazon ElastiCache (Redis OSS) で、適切に設定されたパラメータグループと、クラスターで有効になっているSLOWLOGログ記録。
  - 必要なパラメータは、エンジンバージョンの互換性が Valkey 7.2 以降、または Redis OSSバージョン 6.0 以降に設定されている場合にのみ使用できます。
  - SLOWLOG ログ記録は、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。
- 〔ベスト] CloudWatch または Kinesis Data Firehose の機能を活用します。
  - 、Logs Insights CloudWatch、 CloudWatchAmazon Simple Notification Services のフィルタリングとアラーム機能を使用して、パフォーマンスモニタリングとイベント通知を実現します。
  - Kinesis Data Firehose のストリーミング機能を使用して、SLOWLOGログを永続的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調整をトリガーしたりできます。
  - JSON またはプレーンTEXT形式がニーズに最も適しているかどうかを確認します。
  - CloudWatch または Kinesis Data Firehose に発行するIAMアクセス許可を付与します。
- [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、コマンドが低速実行コマンドとしてログに記録されるまでに、Valkey または Redis OSS エンジン内で コマンドが実行され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、任意の時点で Valkey または Redis OSSメモリにキャプチャされる低速実行コマンドの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工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。SLOWLOG コマンドを使用して valkey-cli から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、この値を増やすことを検討してください。これにより、より多くのコマンドを Valkey または Redis OSSメモリに残すことができます。

  CloudWatch Logs または Kinesis Data Firehose のいずれかにSLOWLOGデータを送信する場合、データは保持され、 ElastiCache システムの外部で分析できるため、実行が遅いコマンドを多数 Valkey または Redis OSSメモリに保存する必要がなくなります。

- [リソース]:
  - ElastiCache (Redis OSS) キャッシュクラスターでスローログを有効にするにはどうすればよいですか？
  - ログ配信
  - Redis OSS固有のパラメータ
  - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
  - Amazon Kinesis Data Firehose

PE8: Auto Scaling はElastiCache クラスターのパフォーマンスを向上させる上でどのように役立ちますか？

質問レベルの紹介: Valkey または Redis OSS 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して、必要なシャードまたは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。

質問レベルの利点: ワークロードの急増を理解し、計画することで、キャッシュのパフォーマンスと事業継続性を強化できます。 ElastiCache （ Redis OSS) Auto Scaling は CPU/Memory 使用率を継続的にモニタリングし、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します。

- 〔必須] ElastiCache (Redis OSS) の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]:

- Valkey および Redis でのクラスターのスケーリング OSS (クラスターモードが有効)
- シャードでの自動スケーリングの使用
- レプリカでの自動スケーリングの使用

- [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]:

  - 自動スケーリングポリシー
  - スケー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に対応する CloudWatch メトリクスを分析して、ターゲット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。

  [リソース]:

  - CloudWatch メトリクスを使用した使用のモニタリング
- [より良い] スケー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]:

  - スケーラブルなターゲットの登録
  - を使用したスケーラブルターゲットの登録 AWS CLI

Amazon ElastiCache Well-Architected レンズコスト最適化の柱

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

- COST 1: リソースに関連するコストをどのように特定して追跡しますか ElastiCache？ 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。
- COST 2: ElastiCache リソースに関連するコストを最適化するために、継続的モニタリングツールをどのように使用しますか？
- COST 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る必要がありますか？ 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。

COST 1: リソースに関連するコストをどのように特定して追跡しますか ElastiCache？ 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、廃止まで作成されたリソースを追跡するためのプロセスとツールを確実に準備しておくと、 に関連するコストを管理するのに役立ちますElastiCache。

質問レベルの利点: 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、 をコンポーネントの 1 つ ElastiCache 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] クラウドセンターオブエクセレンス () を設立憲章の 1 つでインスティテュートし、組織の ElastiCache 使用状況に関するメトリクスを定義、追跡、および実行します。CCoEと 関数CCoE が存在する場合は、 に関連するコストを読み取って追跡する方法を理解している必要がありますElastiCache。リソースが作成されたら、IAMロールとポリシーを使用して、特定のチームや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別データにおける主要な ElastiCache 使用状況について、毎月定期的に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、リーダーシップチームからの非独占的な表現を持つ部門横断的なチームで構成されます。

[リソース]:

- Cloud Center of Excellence の作成
- Amazon ElastiCache の料金

- [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 コスト管理を使用して、 AWS コストと使用量を経時的に視覚化、理解、管理します。
  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 コストを詳細なレベルで追跡します。コスト配分タグを有効にすると、 はコスト配分タグ AWS を使用してコスト配分レポートのリソースコストを整理し、 AWS コストの分類と追跡を容易にします。 AWS は、AWS 生成されたタグとユーザー定義タグの 2 種類のコスト配分タグを提供します。 は、生成されたタグ AWS を定義、作成、適用 AWS し、ユーザー定義のタグを定義、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、独自のコスト構造を反映するように AWS 請求書を整理します。Amazon のリソースにコスト配分タグを追加すると ElastiCache、リソースタグ値別に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 コスト配分タグの使用
  - 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] 組織全体に到達するメトリクスに ElastiCache コストを関連付けます。

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 メトリクスを使用した使用のモニタリング

- 〔良い] up-to-dateを使用するワークロード全体のメトリクスとコストに関するアーキテクチャと運用上の可視性を維持します ElastiCache。
  1. ソリューションエコシステム全体を理解し、クライアントから API Gateway、Redshift、QuickSight レポートツール (例) まで、テクノロジーセット内の AWS サービスの完全なエコシステムの一部になる ElastiCache 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、アプリケーションの運用および機能 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. 可視性、説明責任、優先順位付け、リソースに対するエグゼクティブサポートは、 の効果的なコスト管理戦略を立てる上で不可欠です ElastiCache。

COST 2: ElastiCache リソースに関連するコストを最適化するために、継続的モニタリングツールをどのように使用しますか？

質問レベルの紹介: ElastiCache 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必要があります。Amazon CloudWatch は、 ElastiCache リソースがニーズに対して過剰または十分に活用されていないかどうかを評価するのに役立つ主要な運用メトリクスを可視化します。コスト最適化の観点からは、オーバープロビジョニングのタイミングを理解し、運用、可用性、

耐障害性、パフォーマンスのニーズを維持しながら、リソースのサイズ ElastiCacheを変更するための適切なメカニズムを開発できる必要があります。

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。オーバーサイズの ElastiCache リソースを長期間にわたって特定し、操作しないようにする必要があります。

- 〔必須] CloudWatch を使用してElastiCache クラスターをモニタリングし、これらのメトリクスが AWS Cost Explorer ダッシュボードにどのように関連しているかを分析します。
  1. ElastiCache は、ホストレベルのメトリクス (CPU使用状況など) 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 (キャッシュ取得やキャッシュミスなど) の両方を提供します。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 パフォーマンスメトリクス (CPUUtilization EngineUtilization、SwapUsage、、および Evictions) は CurrConnections、スケールアップ/スケールダウン (大規模/小規模のキャッシュノードタイプを使用) またはイン/アウト (多かれ少なかれシャードを追加) する必要があることを示している場合があります。スケーリング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 メトリクスを使用した使用のモニタリング
  - モニタリングすべきメトリクス
  - Amazon ElastiCache の料金
- [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. では ElastiCache、バックアップは Amazon S3 に保存され、耐久性のあるストレージを提供します。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. 予約済みインスタンスの要件を見積も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 クラスターを操作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。

COST 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る必要がありますか？ 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べる必要があります。

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変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使用可能なSSDストレージデータ階層化インスタンスにより、インスタンスあたりの総データ容量が大幅に増加します。

- [必須] データ階層化インスタンスの制限を理解する
  1. ElastiCache (Redis OSS) クラスターでのみ使用できます。
  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. ElastiCache (Redis OSS) バージョン 6.2 以降のみがサポートされています
  4. 大きな項目は にスワップアウトされませんSSD。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 の料金

- [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 にスワップアウトされるため、アクセスにはわずかなパフォーマンスコストがかかりますSSD。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]:

  - 制約事項
- [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. データ階層化以外のインスタンスを使用して ElastiCache クラスターを操作する必要がある場合があります。これは、要件をよりよく理解するまでです。
  2. ElastiCache クラスターのデータ使用状況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 の料金

を使用した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 ElastiCache

以下のトピックでは、 の使用時に発生する可能性のあるエラーや問題のトラブルシューティングに関するアドバイスを提供します ElastiCache。ここに記載されていない問題が見つかった場合は、このページの [フィードバック] ボタンを使用して報告できます。

トラブルシューティングに関するアドバイスと一般的なサポートに関する質問への回答の詳細については、AWS ナレッジセンターを参照してください。

トピック

- 接続の問題
- Valkey または Redis OSS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問題のトラブルシューティング
- 永続的な接続の問題
- 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. の使用TLS: ElastiCache エンドポイントへの接続時にハング接続が発生している場合は、クライアントTLSで 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 有効なキャッシュ への接続について説明します。
2. VPC： ElastiCache caches には、 内からのみアクセスできますVPC。キャッシュにアクセスするEC2インスタンスと ElastiCache キャッシュが同じ に作成されていることを確認しますVPC。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 間の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 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、こちらをご覧ください。

   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 と 6380、セルフ設計の場合は 6379) へのアクセスを許可していることを確認します。 はこれらのポート ElastiCache を使用して Valkey または Redis OSS 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。

接続が引き続き困難な場合は、他の手順永続的な接続の問題については、「」を参照してください。

Valkey または Redis OSSクライアントエラー

ElastiCache Serverless は、Valkey または Redis OSSクラスターモードプロトコルをサポートするクライアントを使用してのみアクセスできます。独自設計のクラスターは、クラスター設定に応じて、どちらのモードでもクライアントからアクセスできます。

クライアントで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 コマンドでCROSSLOTエラーまたはエラーが発生した場合、クラスタープロトコルをサポートしていない Valkey または Redis OSS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、Valkey または Redis OSSクラスタープロトコルをサポートする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) OSSで Valkey または 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。念のため、 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。

Valkey または Redis OSS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高いと思われる場合は、 SuccessfulReadRequestLatencyと 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することができます。これらのメトリクスは、 ElastiCache サーバーレスの内部にある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。

クライアント側のレイテンシーのトラブルシューティング

クライアント側でレイテンシーが増加しても、サーバー側のレイテンシーを測定する CloudWatch SuccessfulReadRequestLatencyおよび SuccessfulWriteRequestLatencyメトリクスの対応する増加が見られない場合は、以下を考慮してください。

- セキュリティグループがポート 6379 および 6380:Serverless へのアクセスを許可していることを確認します ElastiCache 。このポートは、プライマリエンドポイントには 6379 ポートを使用し、リーダーエンドポイントには 6380 ポートを使用します。一部のクライアントは、アプリケーションがレプリカからの読み取り機能を使用していない場合でも、新しい接続ごとに両方のポートへの接続を確立します。セキュリティグループが両方のポートへのインバウンドアクセスを許可しない場合、接続確立に時間がかかることがあります。ポートアクセスの設定方法については、

  ElastiCache 自己設計型クラスターは Valkey コマンドと Redis OSS 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 Valkey または Redis OSS 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。

  ElastiCache (Memcached) は 11211 および 11212 ポートを使用して Memcached コマンドを受け入れます。EC2 インスタンスから Memcached 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 のセキュリティグループを選択しますVPC。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 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. [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. Valkey または Redis を使用している場合はOSS、ポート範囲ボックスに と入力します6379。
Memcached を使用している場合は、Port range ボックスに と入力します11211。
e. ソースボックスで、ポート範囲 (0.0.0.0/0) がある任意の場所を選択して、Amazon 内で起動する Amazon EC2インスタンスをキャッシュVPCに接続できるようにします。
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. タイプ 列で、カスタムTCPルールを選択します。
h. ElastiCache (Redis OSS) を使用している場合は、Port range ボックスに と入力します6380。
ElastiCache (Memcached) を使用している場合は、Port range ボックスに と入力します11212。
i. ソースボックスで、ポート範囲 (0.0.0.0/0) がある任意の場所を選択して、Amazon 内で起動する Amazon EC2インスタンスをキャッシュVPCに接続できるようにします。
j. [保存] を選択します。

「」を参照してください。

サーバー側のレイテンシーのトラブルシューティング

一部の変動や時折の急増は、懸念の原因とはなりません。ただし、Average統計が急増し、持続する場合は、 AWS Health Dashboard と Personal Health Dashboard で詳細を確認してください。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ために、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、Valkey または Redis OSS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、読み取りをスケーリングし、レイテンシーを低く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ーゾーン (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最終的なデータの整合性を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取りを試みると、アプリケーションが古いデータを受信することがあります。

- アプリケーションがキャッシュAZsと同じ 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AZsと同じ 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ことがあります。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 にデプロイされていることを確認しますAZs。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有効なTCP接続を介して行われます。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、最初のリクエストのレイテンシーが通常よりも長くなります。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提供します。このため、接続プーリングを使用するか、既存の Valkey または Redis OSS接続を再利用することを検討する必要があります。
- スケーリング速度: ElastiCache Serverless は、リクエストレートの増加に応じて自動的にスケーリングされます。 ElastiCache サーバーレスがスケーリングする速度よりも速く、リクエストレートが急激に大きく増加すると、レイテンシーがしばらく長くなる可能性があります。 ElastiCache サーバーレスは通常、サポートされるリクエストレートをすばやく増加させ、リクエストレートを 2 倍にするには最大 10 ~ 12 分かかります。
- 長時間実行されているコマンドの検査: Lua スクリプトや大きなデータ構造のOSSコマンドなど、一部の Valkey または Redis コマンドは長時間実行されることがあります。これらのコマンドを識別するには、 コマンドレベルのメトリクス ElastiCache を発行します。ElastiCache Serverless 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参照してください。
- キーとリクエストの均一な分散: Valkey と Redis ElastiCache ではOSS、スロットあたりのキーまたはリクエストの分散が不均等になると、ホットスロットが発生し、レイテンシーが長くなる可能性があります。 ElastiCache Serverless は、シンプルな 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、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た場合は、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と分散システムでは、さまざまなサービスコンポーネントによってAPIコールが処理されるレート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、スパイクがスムーズになり、コンポーネントのスループットの不一致が制御され、予期しない運用イベントが発生したとき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 Valkey または Redis OSSクライアントはスロットリングされたリクエストに対して再試行を組み込みます。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が、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 Serverless は、より多くのデータを取り込み、リクエストレートを引き上げるたびに自動的にスケーリングされます。アプリケーションが ElastiCache Serverless のスケーリング速度よりも速くスケーリングされると、リクエストがスロットリングされ、 ElastiCache Serverless はワークロードに合わせてスケーリングされます。 ElastiCache Serverless は通常、ストレージサイズをすばやく増やすことができ、キャッシュ内のストレージサイズを 2 倍にするには最大 10 ~ 12 分かかります。
- キーとリクエストの均一な分散: Valkey または Redis ElastiCache ではOSS、スロットあたりのキーまたはリクエストの分散が不均等になると、ホットスロットが発生する可能性があります。単一のスロットへのリクエストレートが 30,000 ECPUs/秒を超える場合、単純な SET/GET コマンドを実行するワークロードでホットスロットを使用すると、リクエストがスロットリングされる可能性があります。
- レプリカから読み取る: アプリケーションで許可されている場合は、「レプリカから読み取る」機能の使用を検討してください。ほとんどの Valkey または Redis OSSクライアントは、「スケールリード」を設定して、リードをレプリカノードに誘導できます。この機能を使用すると、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 Serverless は、レプリカリクエストから読み取られたノードを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自動的にルーティングするため、レイテンシーが低くなります。レプリカからの読み取りが有効になって

いる場合、シンプルな ECPUs/ コマンドのワークロードに対して、1 つのスロットで最大 90,000 SET/GET秒を達成できます。

永続的な接続の問題

での永続的な接続問題のトラブルシューティング中に、次の項目を検証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 ACLs
- 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 使用状況
- 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) と ElastiCache 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、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。クライアントまたは ElastiCache 側で制限がヒット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。

クライアントと ElastiCache クラスターに同時に単一のセキュリティグループを割り当てるか、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ができます。

どちらの場合も、送信元からの ElastiCache ポートのTCPアウトバウンドトラフィックと、同じポートのインバウンドトラフィックを に許可する必要があります ElastiCache。デフォルトのポートは、Memcached の場合は 11211、Valkey または Redis の場合は 6379 ですOSS。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 の ElastiCache クラスターにアクセスするためのアクセスパターン VPC」を参照してください。

ネットワーク ACLs

ネットワークアクセスコントロールリスト (ACLs) 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 ACLs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。特に同じサブネット内にある場合は、 ElastiCache とクライアントリソースに同じ ACL を割り当てることができます。

デフォルトでは、ネットワークはすべてのトラフィックACLs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、トラフィックと一致する最小数を持つルールは、ルールを許可または拒否します。Valkey または Redis OSSトラフィックを許可する最小設定は次のとおりです。

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
- プロトコル: TCP
- [Port Range]: 1024 ~ 65535
- ソース: 0.0.0.0/0 (または ElastiCache クラスターサブネットの個別ルールを作成する)
- [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

- プロトコル: TCP
- [Port Range]: 6379
- ソース: 0.0.0.0/0 (または ElastiCache クラスターサブネット。 特定の を使用すると、クラスターのフェイルオーバーやスケーリング時に問題が発生するIPs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
- プロトコル: TCP
- [Port Range]: 6379
- ソース: 0.0.0.0/0 (または ElastiCache クラスターサブネットの個別ルールを作成する)
- [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
- プロトコル: TCP
- [Port Range]: 1024 ~ 65535
- ソース: 0.0.0.0/0 (または ElastiCache クラスターサブネット。 特定の を使用すると、クラスターのフェイルオーバーやスケーリング時に問題が発生するIPs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ACLs」を参照してください。

ルートテーブル

ネットワーク と同様にACLs、各サブネットには異なるルートテーブルを含めることができます。クライアントと ElastiCache クラスターが異なるサブネットにある場合は、それらのルートテーブルが互いに到達できることを確認してください。

複数の 、動的ルーティングVPCs、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では、トラブルシューティングが困難になる場合があります。「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 は、DNS名前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、 VPCDNSサービスを使用しない) では、 ElastiCacheが提供するDNS名前が認識されない場合があります。 (次digに示すように) や などのシステムツールを使用して、システムが ElastiCache エンドポイントを正常に解決できることを確認します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 サービスを通じ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.
1.2.3.4
```

サーバー側の診断に関する問題の特定

CloudWatch ElastiCache エンジンからのメトリクスとランタイム情報は、接続問題の潜在的な原因を特定するための一般的なソースまたは情報です。適切な分析は、通常、次の項目から始まります。

- CPU 使用状況: Valkey と Redis OSS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、 はメトリクスCPUUtilizationとElastiCache を提供しますEngineCPUUtilization。 EngineCPUUtilizationは、Valkey または Redis OSSプロセス専用のCPU使用率と、すべての のCPUUtilization使用状況を提供しますvCPUs。複数の vCPU を持つノードは通常、 CPUUtilizationと の値が異なるためEngineCPUUtilization、2 つ目は通常高くなります。高 は、リクエスト数の増加や、完了までにかなりCPUの時間がかかる複雑な操作が原因EngineCPUUtilization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ら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 と相関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取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、 ElastiCache 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。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ートポロジー、レプリカノードを自動的に検出し、READONLYValkey または Redis OSS コマンドを使用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できます。
- 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、大幅なCPUオーバーヘッドを意味します。さらに、新しい接続の作成に必要な TCP3 方向ハンドシェイクは、全体的な応答時間に悪影響を及ぼします。

    1 分NewConnectionsあたり数千の ElastiCache ノードは、接続が作成され、いくつかのコマンドによって使用されることを示しますが、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。Valkey と Redis は、少数の で最適なパフォーマンスOSSを提供しますcurrConnections。数十から数百の順序 currConnection で保持すると、クライアントバッファや接続に役立つCPUサイクルなどの個々の接続をサポートするリソースの使用が最小限に抑えられます。

- ネットワークスループット:
  - 帯域幅： ElastiCache ノードのネットワーク帯域幅がノードサイズに比例することを確認します。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。例えば、小さなリクエストのレートが高いアプリケーションは、ネットワークスループットよりもCPU使用量に影響する傾向があり、キーが大きいほどネットワーク使用率が向上します。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。書き込み集約型ワークロードのベストプラクティスは ElastiCache 、 (Redis OSS) を使用してクラスターモードを有効にし、複数のシャード間で書き込みのバランスを取るか、より多くのネットワーク機能を持つノードタイプにスケールアップすることです。
  - および CloudWatchmetrics `NetworkBytesIn`は、ノードに出入りするデータ量をそれぞれ`NetworkBytesOut`提供します。 `ReplicationBytes`は、データレプリケーション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 OSS コマンドは 1 つのスレッドで提供されます。つまり、リクエストは順番に提供されます。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 OSSは「ビッグO」表記を使用して、コマンドの複雑さを記述します。

  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ディングします。

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 回の繰り返しを必要とします。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 100,0000 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イープを終了するまで、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。すべてのコマンドとその時間の複雑さのリストを確認するには、「Valkey コマンド」と「Redis OSS コマンド」を参照してください。

潜在的な問題の例:

- Lua スクリプト: Valkey と Redis OSSは埋め込み Lua インタープリタを提供し、サーバー側のスクリプトの実行を許可します。Valkey および Redis の Lua スクリプトOSSはエンジンレベルで実行され、定義上原子です。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を実行することはできません。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。では ElastiCache、Lua スクリプトの実行時間は 5 秒に制限されます。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OSS。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、ハッシュ (複数のサブ項目を含むデータ構造) が含まれている場合、かなりCPUの時間がかかります。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。の代替DEL手段は です。これは UNLINKRedis OSS4 以降に利用可能な非同期コマンドです。 はDEL可能な限り よりも優先UNLINKする必要があります。 ElastiCache (Redis OSS) 6x 以降、 lazyfree-lazy-user-delパラメータは、有効UNLINKになっているときにDELコマンドの動作を のようにします。詳細については、「Redis OSS 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 OSSには、同様の動作をするコマンドが他にも多数用意されています。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使用状況に影響します。

  CPU 使用率だけでは接続の問題は発生しませんが、複数のキーで 1 つ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、全体的なCPU使用率が高くな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。
- 複数のデータ型で動作するコマンド: Redis は、データ型に関係なく、1 つ以上のキーで動作するコマンドOSSも提供します。 ElastiCache (Redis OSS) は、このようなコマンドをモニタリングKeyBasedCmdsするためのメトリクスを提供します。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。
- その他のビッグデータ構造: ハッシュに加えて、他のデータ構造も大量CPU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、「Valkey コマンドと Redis OSS コマンド」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 ACLs、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、VPCReachability Analyzer とシステムツールを使用して接続を検証できます。

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、 で使用可能な ElastiCache ノードの 1 つの ENI ID (Elastic Network Interface Identification) が必要ですVPC。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 に移動しますか?#NIC：
2. インターフェイスリストを ElastiCache クラスター名または以前にDNS検証した IP アドレスで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イスが表示されている場合は、説明を確認して正しい ElastiCache クラスターに属することを確認し、その 1 つを選択します。
4. 次のステップに進みます。
5. https://console.aws.amazon.com/vpc/自宅で分析パスを作成しますか?#ReachabilityAnalyzer で、次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、または awsvpc ネットワークECSを使用する AWS Fargate Amazon などの別のサービスを使用している場合はネットワークインターフェイスで実行されている場合 AWS Lambda、インスタンスを選択します) 、およびそれぞれのリソース ID (EC2インスタンスまたは ENI ID)。
   - 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 : ElastiCache (Redis OSS) には 6379、 ElastiCache (Memcached) には 11211 を指定します。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。
   - プロトコル : 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TCPの接続を検証するには: Amazon Linux では、 Npingがパッケージで利用可能nmapで、 ElastiCache ポートTCPの接続をテストできるだけでなく、接続を確立するためのネットワーク往復時間も指定できます。これを使用して、次に示すように、ネットワーク接続とElastiCache 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 つのプローブをそれらの間で 1 秒の遅延で送信します。オプション「-c」を使用してプローブ数を増やし、「—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。

テストがnping失敗し、VPCReachability Analyzer テストに合格した場合は、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、またはオペレーティングシステムレベルでの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 が別のものを返します。エラーの原因を https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS で確認してください。

すべてのインフラストラクチャとオペレーティングシステムのテストに合格してもアプリケーションが に接続できない場合は ElastiCache、アプリケーション設定が ElastiCache 設定に準拠しているかどうかを確認します。よくある間違いは次のとおりです。

- アプリケーションは ElastiCache クラスターモードをサポートしておらず、クラスターモードが有効になってい ElastiCache ます。

- アプリケーションは TLS/ をサポートしておらずSSL、転送中の暗号化が有効になっていElastiCache ます。
- アプリケーションは TLS/SSL をサポートしていますが、適切な設定フラグや信頼できる認証機関はありません。

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。各 ElastiCache ノードでは、すべてのクライアント間で最大 65,000 の同時接続が可能です。この制限は、 のCurrConnectionsメトリクスを通じてモニタリングできます CloudWatch。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、28231 接続は同じ送信元から同じ送信先 IP (ElastiCache ノード) とポートに許可されます。次のコマンドは、特定の ElastiCache ノード (IP 1.2.3.4) にいくつの接続が存在するかを示します。

  ```
  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の制限: CloudWatch Redis の次のメトリクスOSSを確認して、ElastiCache ノードでヒットする可能性のあるネットワーク制限を特定します。
  - 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

力、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)、レプリカの数を減らす、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えています。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。

プラトーが長期間観察されると、レプリケーションの遅延、キャッシュに使用されるバイトの増加、空きメモリのドロップ、スワップの増加、CPU使用量の増加が続く可能性があります。Amazon EC2インスタンスには、ENAドライバーメトリクス を通じて追跡できるネットワーク制限もあります。ネットワークサポートが強化され、ENAドライバーが 2.2.10 以降の Linux インスタンスは、 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 使用状況

CPU 使用状況メトリクスは調査の出発点であり、以下の項目は ElastiCache 、側で発生する可能性のある問題を絞り込むのに役立ちます。

- Redis OSS SlowLogs: ElastiCache デフォルト設定では、完了までに 10 ミリ秒以上かかった最後の 128 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 に発生しUTC、完了までに 4.8 秒 (4.823 ミリ秒) かかり、クライアント 1.2.3.4 からリクエストされたKEYSコマンドによって引き起こされました。

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 を使用する Windows の場合 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) 時間の複雑さを持つ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 OSS) は、さまざまなクラスのコマンドの平均レイテンシーをモニタリングするための CloudWatch メトリクスを提供します。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 Script コマンド、、evalevalsha、
  - GeoSpatialBasedCmdsLatency: geodist, geohash, geopos, georadius, georadiusbymember, geoadd;
  - GetTypeCmdsLatency: データ型に関係なくコマンドを読み取る
  - HashBasedCmdsLatency: hexists, hget, hgetall, hkeys, hlen, hmget, hvals, hstrlen, hdel, hincrby, hincrbyfloat, hmset, hset, hsetnx;
  - HyperLogLogBasedCmdsLatency: pfselftest, pfcount, pfdebug, pfadd, pfmerge;
  - KeyBasedCmdsLatency: 、dump、、exists、、、keysobjectpttl、、、、randomkey、、ttlty unlink
  - ListBasedCmdsLatency: lindex、llen、lrange、blpop、brpop、brpoplpush、linsert、lpop、lpush、lpushx、lrem、lset、ltrim、rp
  - PubSubBasedCmdsLatency: psubscribe、publish、pubsub、punsubscribe、subscribe、unsubscribe;
  - 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む
  - 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 OSSランタイムコマンド：
  - info commandstats: Redis OSS エンジンの起動後に実行されたコマンドのリスト、累積実行数、合計実行時間、およびコマンドあたり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。
- より前のバックアップとレプリケーション: ElastiCache (Redis OSS) バージョンでは、フォークプロセス2.8.22を使用してバックアップを作成し、レプリカとの完全な同期を処理します。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 OSS2.8.22 以降、 はフォークレスバックアップとレプリケーションメソッド AWS を導入しました。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がタイムアウト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

デフォルトの ElastiCache (Redis OSS) 設定では、クライアント接続は無期限に確立されます。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、クライアントまたは ElastiCache 側で飽和が発生する可能性があります。クライアント側から即時修正ができない場合、一部の管理者は ElastiCache パラメータグループに「タイムアウト」値を設定します。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中にリクエストを送信しない場合、Redis OSS エンジンは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、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。デフォルトの ElastiCache (Redis OSS) 設定では、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ます。メモリが非常に低い状況では、 ElastiCache (Redis OSS) は、メモリを解放し、クラスターの状態を維持するために、大きな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: 単一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制限。で定義された秒単位の時間を超えてソフト制限を超えたままの場合、 client-output-buffer-limitnormal-soft-seconds またはハード制限に達した場合、接続は終了します。
  - client-output-buffer-limit-normal-soft-seconds: client-output-buffer-limit-normal-soft-limit; を超える接続に許可される時間
  - client-output-buffer-limit-normal-hard-limit: この制限に達する接続はすぐに終了します。

  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は、 へのリクエストにも影響します ElastiCache。EC2 断続的な接続やタイムアウトの問題をトラブルシューティングする際には、インスタンスと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CPUの使用状況: CPUが飽和していないか、100% に近いことを確認してください。履歴分析は を介して行うことができますが CloudWatch、データポイントの粒度は 1 分 (詳細モニタリングが有効になっている) または 5 分であることに注意してください。
  - バーストインスタンス を使用する場合はEC2、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。この情報は、 CPUCreditBalance CloudWatch メトリクスで入手できます。
  - CPU 使用率が高い期間が短いと、 の使用率を 100% 反映せずにタイムアウトが発生す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 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: NTPを介したインスタンスメタデータへの過剰なリクエストが原因でドロップされたパケットの数VPCDNS。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。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、PPS制限に達する可能性があります。 ElastiCache トラフィックは、 MGETではなく複数のオペレーションを同時に実行するパイプラインまたはコマンドを介してネットワークパケットをより適切に使用するように最適化できます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および Wireshark (コマンドライン上のサメ) は、リクエストがネットワークを移動し、 ElastiCache エンジンをヒットしてリターンを得るのにかかった時間を把握する

ための便利なツールです。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、TCP3 方向ハンドシェイクが 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。で同じ Redis OSS PING コマンドを使用するとstrace、次のようになります。

```
$ 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"], 0x7fffede7cc38 /* 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 をインスタンス化し、名前解決 (697712 から 717890 まで) を行うには、約 20 ミリ秒というかなりの時間がかかりました。その後、TCPソケットの作成に 2 ミリ秒 (745659 から 747858)、リクエストの応答を送信して受信するには 0.4 ミリ秒 (747858 から 748330) が必要でした。

関連トピック

- the section called “ベストプラクティスとキャッシュ戦略”

Amazon のセキュリティ ElastiCache

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、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 とユーザーの間で責任を共有します。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 は、安全に使用できるサービス AWS も提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon 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による対象範囲内のサービス」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、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共有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、セキュリティとコンプライアンスの目的を満たす ElastiCache ように Amazon を設定する方法を示します。また、Amazon ElastiCache リソースのモニタリングと保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon でのデータ保護 ElastiCache
- 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 のレジリエンス ElastiCache
- のインフラストラクチャセキュリティ AWS ElastiCache
- のサービス更新 ElastiCache
- 一般的な脆弱性と露出 (CVE): で対処されるセキュリティの脆弱性 ElastiCache

Amazon でのデータ保護 ElastiCache

責任 AWS 共有モデル 、 AWS ElastiCache () のデータ保護に適用されますElastiCache。このモデルで説明されているように、 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があります。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 サービスのセキュリティ設定および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FAQ」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、 AWS Identity and Access Management () を使用して個々のアカウントを設定することをお勧めしますIAM。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。
- を使用して AWS リソースと通信TLSします。
- で APIとユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 サービス内のすべての AWS デフォルトのセキュリティコントロールを使用します。
- 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。これには、コンソール、、 ElastiCache または を使用して または他の AWS API AWS CLIサービスを使用する場合も含まれます AWS SDKs。入力したデータ ElastiCache やその他のサービスが、診断ログに取り込まれる可能性があります。URL を外部サーバーに提供するときは、そのサーバーへのリクエストを検証URLするために認証情報を に含めないでください。

トピック

- Amazon のデータセキュリティ ElastiCache

Amazon のデータセキュリティ ElastiCache

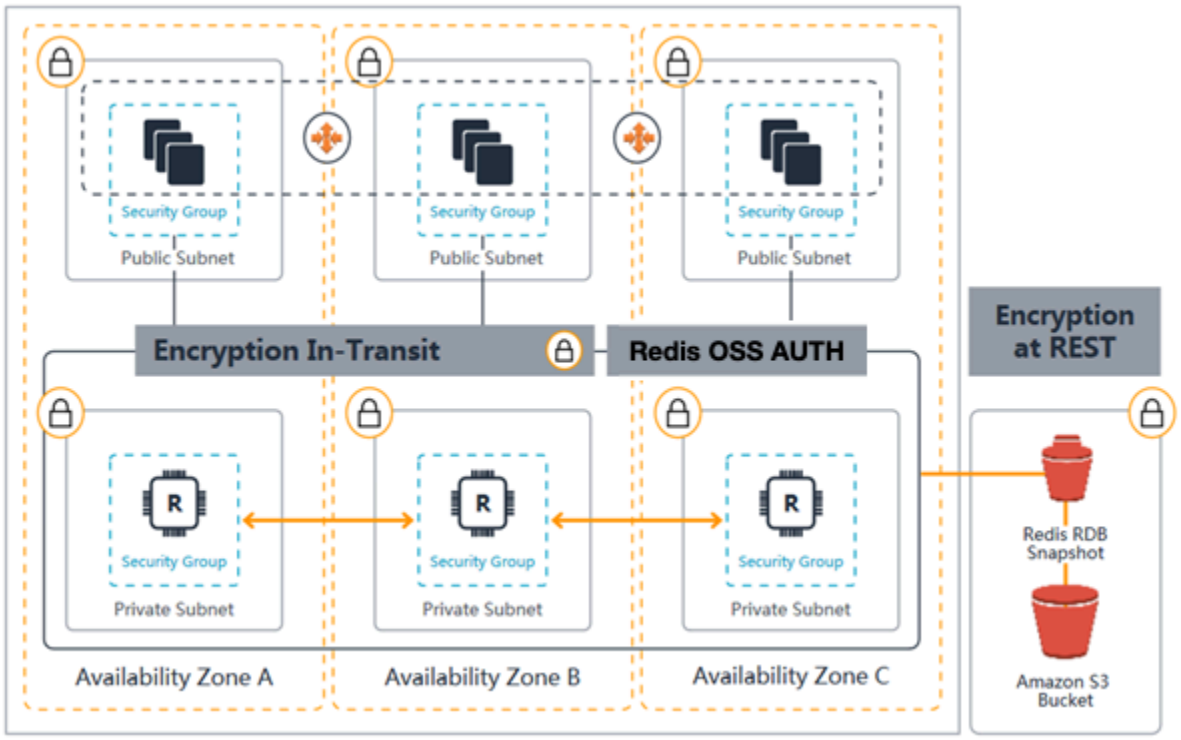
データを安全に保護するために、Amazon ElastiCache と Amazon EC2はサーバー上のデータの不正アクセスから保護するメカニズムを提供しています。

Amazon ElastiCache (Memcached) は、Memcached バージョン 1.6.12 以降を実行しているキャッシュ上のデータの暗号化機能を提供します。

Amazon ElastiCache with Valkey と Redis OSSは、Valkey 7.2 以降、および Redis OSSバージョン 3.2.6 (用にスケジュールされていますEOL。Redis OSSバージョンの有効期限スケジュール を参照) 、4.0.10 以降を実行するキャッシュ上のデータの暗号化機能を提供します。Amazon は、 IAM または Valkey と Redis のいずれかを使用してユーザーを認証しOSSAUTH、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オペレーションを承認 ElastiCache することもできますRBAC。

- 転送時の暗号化では、ある場所から別の場所に移動するデータ (クラスターのノード間、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なわれます。
- 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。

ElastiCache は、 IAM および Valkey と Redis OSS AUTH コマンドを使用したユーザーの認証、およびロールベースのアクセスコントロール () を使用したユーザーオペレーションの承認をサポートしますRBAC。



ElastiCache Valkey と Redis OSS セキュリティ図

トピック

- ElastiCache 転送中の暗号化 (TLS)
- の保管時の暗号化 ElastiCache
- 認証と認可

ElastiCache 転送中の暗号化 (TLS)

データを安全に保護するために、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ています。転送中の暗号化機能を提供することで、 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 ElastiCache を提供します。

すべての Valkey または Redis OSSサーバーレスキャッシュでは、転送中の暗号化が有効になっています。独自設計のクラスターでは、レプリケーショングループの作成時に パラメータ`TransitEncryptionEnabled`を `true` (CLI：`--transit-encryption-enabled`) に設定することで、レプリケーショングループの転送中の暗号化を有効にできます。これは、、 AWS Management Console、 AWS CLIまたは を使用してレプリケーショングループを作成するかどうかにかかわらず行うことができます ElastiCache API。

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。自己設計型クラスターの場合、 `true` (CLI: `--transit-encryption-enabled`) オペレーションを使用してキャッシュクラスターを作成するときに、 パラメータ`TransitEncryptionEnabled`を `CreateCacheCluster` (CLI: `create-cache-cluster`) に設定することで、キャッシュクラスターで転送中の暗号化を有効にできます。

トピック

- 転送時の暗号化の概要
- 転送中の暗号化条件 (Valkey と Redis OSS)
- 転送中の暗号化条件 (Memcached)
- 転送時の暗号化のベストプラクティス
- その他の Valkey および Redis OSSオプション
- Memcached の転送中の暗号化の有効化
- 転送時の暗号化を有効にする
- valkey-cli を使用した転送中の暗号化による ElastiCache (Valkey) または Amazon ElastiCache (Redis OSS) への接続

- Python を使用して独自設計の Redis OSS クラスターで転送中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
- Openssl (Memcached) を使用した転送中の暗号化で有効化されたノードへの接続
- Java を使用した TLS Memcached クライアントの作成
- を使用した TLS Memcached クライアントの作成 PHP

転送時の暗号化の概要

Amazon ElastiCache 転送中の暗号化は、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高める機能です。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 転送中の暗号化では、次の機能を実装します。

- 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 — Valkey と Redis OSSAUTH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。

転送中の暗号化条件 (Valkey と Redis OSS)

Amazon ElastiCache の転送中の暗号化に関する以下の制約事項は、独自設計のクラスター実装を計画する際に考慮する必要があります。

- 転送中の暗号化は、Valkey 7.2 以降、および Redis OSSバージョン 3.2.6、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ています。
- 既存のクラスターの転送中の暗号化設定を変更することは、Valkey 7.2 以降、および Redis OSS バージョン 7 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ています。
- 転送中の暗号化は、Amazon 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ていますVPC。
- 転送中の暗号化はM1, 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、クライアント設定で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 以降のすべての AWS リージョンで廃止されます。ElastiCache は、2025 TLS 年 5 月 8 日まで 1.0 と 1.1 を引き続きサポートします。お客様は、その日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。

転送中の暗号化条件 (Memcached)

Amazon ElastiCache の転送中の暗号化に関する以下の制約事項は、独自設計のクラスター実装を計画する際に考慮する必要があります。

- 転送時の暗号化は、Memcached バージョン 1.6.12 以降を実行するクラスターでサポートされます。
- 転送中の暗号化は、Transport Layer Security (TLS) バージョン 1.2 および 1.3 をサポートしています。
- 転送中の暗号化は、Amazon で実行されているクラスターでのみサポートされますVPC。
- 転送中の暗号化は、M1, M2, M3, R3, T2 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。
- 転送時の暗号化は、クラスターの作成時にのみクラスターで有効にできます。クラスターを変更して転送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。
- キャッシュクライアントがTLS接続をサポートし、クライアント設定で有効にしていることを確認します。

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コストがかかる可能性があるため、TLS接続を保持することで、転送中の暗号化のパフォーマンスへの影響を軽減できます。

その他の Valkey および Redis OSSオプション

Valkey および Redis で使用できるオプションの詳細についてはOSS、以下のリンクを参照してください。

- の保管時の暗号化 ElastiCache
- Valkey および Redis OSS 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)
- Amazon VPCs と ElastiCache セキュリティ
- Amazon の Identity and Access Management ElastiCache

Memcached の転送中の暗号化の有効化

AWS マネジメントコンソールを使用して Memcached クラスターの作成時に転送時の暗号化を有効にするには、以下のように選択します。

- エンジンとして Memcached を選択します。
- エンジンバージョン 1.6.12 以降。
- [Encryption in transit] (転送時の暗号化) で、[Enable] (有効化) を選択します。

プロセスについては、 step-by-step「」を参照してくださいValkey または Redis 用のクラスターの作成 OSS。

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。自己設計型クラスターでは、 AWS Management Console、、 AWS CLIまたは ElastiCache を使用して転送中の暗号化を有効にできますAPI。

を使用して転送中の暗号化を有効にする AWS Management Console

を使用して、新しい独自設計クラスターの転送時の暗号化を有効にする AWS Management Console

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。

プロセスについては step-by-step、以下を参照してください。

- Valkey (クラスターモードが無効) クラスターの作成 (コンソール)
- Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)

を使用して既存の独自設計クラスターの転送時の暗号化を有効にする AWS Management Console

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、Valkey または Redis OSS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。暗号化された接続を使用するようにすべての Valkey または Redis OSSクライアントを移行したら、クラスター設定を変更してトランジット暗号化モードを に設定することができます`required`。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。

Transit 暗号化モードを優先に設定する

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインにリストされているリソースから ElastiCache、左側にある Valkey キャッシュまたは Redis OSSキャッシュを選択します。
3. 更新するキャッシュを選択します。
4. [アクション] ドロップダウンを選択してから、[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。

暗号化された接続を使用するようにすべての Valkey または Redis OSSクライアントを移行した後：

Transit 暗号化モードを必須に設定する

1. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインにリストされているリソースから ElastiCache、左側にある Valkey キャッシュまたは Redis OSSキャッシュを選択します。
3. 更新する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、[変更のプレビュー] を選択します。

を使用して転送中の暗号化を有効にする AWS CLI

を使用して Valkey または Redis OSSレプリケーショングループを作成するときに転送時の暗号化を有効にするには AWS CLI、 パラメータ を使用しますtransit-encryption-enabled。

Valkey または Redis (OSSクラスターモードが無効) (CLI) 用の新しい自己設計型クラスターで転送中の暗号化を有効にする

AWS CLI オペレーションcreate-replication-groupと次のパラメータを使用して、転送中の暗号化が有効になっているレプリカを持つ Valkey または Redis OSS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**— valkeyまたは である必要がありますredis。
- **--engine-version**— エンジンが Redis の場合OSS、 は 3.2.6、4.0.10 以降である必要があります。
- **--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。

- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する (AWS CLI)
- create-replication-group

Valkey または Redis (OSSクラスターモードが有効) (CLI) 用の新しい自己設計型クラスターで転送中の暗号化を有効にする

AWS CLI オペレーション`create-replication-group`と次のパラメータを使用して、転送中の暗号化が有効になっている Valkey または Redis OSS (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`**— `valkey`または である必要がありま`redis`す。
- **`--engine-version`**— エンジンが Redis の場合OSS、 は 3.2.6、4.0.10 以降である必要があります。
- **`--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。

- Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成する (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、Valkey または Redis OSS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。暗号化された接続を使用するようにすべての Valkey または Redis OSSクライアントを移行したら、クラスター設定を変更してトランジット暗号化モードを に設定することができます`required`。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、転送中の暗号化が無効になっている Valkey または Redis OSS (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 `preferred`を に設定します transit-encryption-mode。
   - **`--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`required`、 を に設定します transit-encryption-mode。
   - **`--transit-encryption-enabled`** — 必須
   - **`--transit-encryption-mode`** — `required` に設定する必要があります。

valkey-cli を使用した転送中の暗号化による ElastiCache (Valkey) または Amazon ElastiCache (Redis OSS) への接続

転送中の暗号化で有効になっている ElastiCache (Redis OSS) キャッシュからデータにアクセスするには、Secure Socket Layer () を使用するクライアントを使用しますSSL。Amazon Linux および Amazon Linux 2 では、TLS/SSL で valkey-cli を使用することもできます。クライアントが をサポートしていない場合はTLS、クライアントホストの `stunnel` コマンドを使用して、Redis OSS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

valkey-cli を使用して、Amazon Linux 2023、Amazon Linux 2、または Amazon Linux で転送中の暗号化が有効になっている Valkey または Redis OSSクラスターに接続するには、次の手順に従います。

1. valkey-cli ユーティリティをダウンロードしてコンパイルします。このユーティリティは Valkey 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、この例に示すものに置き換えます。

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

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

   Amazon Linux 2

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

   ```
   sudo yum -y install openssl-devel gcc
   wget https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   tar xvzf valkey-7.2.6.tar.gz
   cd valkey-7.2.6
   make distclean
   make valkey-cli BUILD_TLS=yes
   sudo install -m 755 src/valkey-cli/usr/local/bin/
   ```

   Amazon Linux

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

   ```
   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
   wget https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   tar xvzf valkey-7.2.6.tar.gz
   cd valkey-7.2.6
   make valkey-cli CC=clang BUILD_TLS=yes
   ```

```
sudo install -m 755 src/valkey-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. valkey-cli ユーティリティをダウンロードしてインストールしたら、オプションの `make-test` コマンドを実行することをお勧めします。
4. 暗号化と認証が有効になっているクラスターに接続するには、次のコマンドを入力します。

```
valkey-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と、 `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valkey-cli。
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

stunnel による暗号化された接続

valkey-cli を使用して、stunnel を使用した転送中の暗号化が有効になっている Redis OSS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、 をインストールしますstunnel。

```
sudo yum install stunnel
```

2. 以下のコマンドを実行して、ファイル`'/etc/stunnel/valkey-cli.conf'`を同時に作成および編集し、以下の提供された出力をテンプレートとして使用して ElastiCache 、(Redis OSS) クラスターエンドポイントを 1 つ以上の接続パラメータに追加します。

```
vi /etc/stunnel/valkey-cli.conf


fips = no
```

```
setuid = root
setgid = root
pid = /var/run/stunnel.pid
debug = 7
delay = yes
options = NO_SSLv2
options = NO_SSLv3
[valkey-cli]
   client = yes
   accept = 127.0.0.1:6379
   connect = primary.ssltest.wif01h.use1.cache.amazonaws.com:6379
[valkey-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 valkey-cli と valkey-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 で Redis のOSS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 ユーザーガイドの「エフェメラルポート」を参照してください。VPC
- connect は Redis OSSサーバーエンドポイントに設定されます。詳細については、「での接続エンドポイントの検索 ElastiCache」を参照してください。

3. stunnel を起動します。

```
sudo stunnel /etc/stunnel/valkey-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 OSSノードに接続します。

   - ElastiCache (Redis OSS) 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、valkey-cli を使用して Amazon Linux の valkey-cli の完全なパスを使用して ElastiCache (Redis OSS) サーバーに接続します。

     ```
     /home/ec2-user/redis-7.2.5/src/valkey-cli -h localhost -p 6379
     ```

     Redis OSSクラスターの作成時にAUTHパスワードが使用された場合、この例では、valkey-cli を使用して Amazon Linux で valkey-cli の完全なパスを使用して Redis OSSサーバーに接続します。

     ```
      /home/ec2-user/redis-7.2.5/src/valkey-cli -h localhost -p 6379 -a my-secret-
     password
     ```

   または

   - ディレクトリを redis-7.2.5 に変更し、以下を実行します。

     ElastiCache (Redis OSS) 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、valkey-cli を使用して Amazon Linux の valkey-cli の完全なパスを使用して ElastiCache (Redis OSS) サーバーに接続します。

     ```
     src/valkey-cli -h localhost -p 6379
     ```

     Redis OSS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、valkey-cli を使用して、Amazon Linux で valkey-cli の完全なパスを使用して Valkey または Redis OSS サーバーに接続します。

     ```
     src/valkey-cli -h localhost -p 6379 -a my-secret-password
     ```

   この例では、Telnet を使用して Valkey Redis OSS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、stunnel プロセスpkillを使用します。

```
sudo pkill stunnel
```

Python を使用して独自設計の Redis OSS クラスターで転送中の暗号化を有効にする

次のガイドでは、転送中の暗号化が無効になっている状態で最初に作成された Redis OSS7.0 クラスターで転送中の暗号化を有効にする方法を示します。TCP および TLSクライアントは、ダウンタイムなしで、このプロセス中に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。次の例のコードは、 ElastiCache Redis OSSクラスターの作成VPCに使用されるのと同じで起動されたEC2インスタンスから処理されました。

> **Note**
>
> - 次の例では、 ElastiCache boto3 SDK を管理オペレーション (クラスターまたはユーザー作成) と redis-py/redis-py-cluster for data handling に使用します。
> - クラスターの変更でオンラインTLS移行を使用するには、少なくとも boto3 バージョン (=~) 1.26.39 を使用する必要がありますAPI。
> - ElastiCache は、Valkey バージョン 7.2 以降または Redis OSSバージョン 7.0 以降のクラスターでのみオンラインTLS移行をサポートします。そのため、7.0 より前のOSS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 OSS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、「Redis とのメジャーバージョンの動作と互換性の違い OSS」を参照してください。

トピック

- ElastiCache Valkey または Redis OSS 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) Valkey または Redis OSS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を作成する
- 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 Valkey または Redis OSS 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などの 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単純な Python 文字列定数を定義しましょうdefault parameter group。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

次に、クラスターの設定を表す単純な Python クラスをいくつか定義します。このクラスには、Valkey バージョンや Redis OSS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含まれます。

```
#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 Valkey または Redis OSSクラスター自体を表す単純な Python クラスをいくつか定義します。このクラスには、クラスターの作成や のクエリなどの ElastiCache 管理オペレーション用に boto3 クライアントを保持するクライアントフィールドがあります ElastiCacheAPI。

```
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 preffered 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) Valkey または Redis OSS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を作成する

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
improt 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

    # unsuccesful 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 client を使用してクラスターを作成します。
2. なしで明確な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TLS。
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、no-TLSto- TLS preferred TLSから preferred への移行をトリガーします。

5. クラスターを TLS に移行する間Preferred、redis-py-clusterTCP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. への移行が完了したら、クラスターTLSPreferredが転送中の暗号化をサポートしていることをアサートします。その後、 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TLS。
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 からTLS必須TLSPreferredへの移行をトリガーします。
9. クラスターをTLS必須に移行している間、 redis-py-clusterTLS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 migartion 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に、適切なDNSレコード処理があることを確認してください。

> **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。エンドポイントを誤って使用すると、Valkey または Redis OSSクライアントが古いエンドポイントと削除されたエンドポイントを使用してクラスターに接続できなくなる可能性があります。

クラスターが推奨されないTLS からTLS推奨されない に移行する間、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、クラスターとは異なる non-TLS-enabled形式のDNSレコードを使用します。 ElastiCache は、クラスターが暗号化モードで設定されているときに両方のDNSレコードを保持します。アプリケーションと他の Valkey または Redis OSSクライアントがそれらを切り替えられるようにすることをお勧めします。DNS レコードでは、TLS移行プロセス中に次の変更が行われます。

転送中の暗号化を有効にするときに発生する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:

- 有効になっていないTLS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のままになります。クラスターがTLS暗号化モード「なし」から「優先」に再設定された場合、ダウンタイムはありません。
- 新しい TLS Valkey または Redis OSSエンドポイントは、クラスターが TLS優先モードに設定されているときに生成されます。これらの新しいエンドポイントは、古いエンドポイントIPsと同じ (非) に解決されますTLS。
- 新しい TLS Valkey または Redis OSS設定エンドポイントは、ElastiCache コンソールと `describe-replication-group` へのレスポンスで公開されますAPI。

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:

- 有効になっていないTLS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ありません。
- ElastiCache コンソールまたは `cluster-configuration-endpoint` から新しい `describe-replication-group` を取得できますAPI。

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になっている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:

- 有効になっていないTLS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のままになります。
- クラスターが TLS `Preferred` モードに設定されていると、新しいTLS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 (非) と同じ IP (複数可) に解決されますTLS。
- 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` へのレスポンスで公開されますAPI。

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:

- 古い非TLS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ありません。
- コンソールまたは から ElastiCache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を取得できますdescribe-replication-groupAPI。

DNS レコードの推奨使用法

CMEクラスターの場合

- アプリケーションのコードのノードごとのDNSレコードではなく、クラスター設定エンドポイントを使用します。ノードごとのDNS名前を直接使用することはお勧めしません。シャードを追加または削除するときに変更される可能性があるためです。
- 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 (上記の (太字) で示すように) でクラスター設定エンドポイントをクエリしDNS、この時点から取得した を使用します。

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 no-preferred から -TLSTLSpreferred に移行すると、古いノード名が削除され、新しいノードDNS名が生成されるため、アプリケーションのコードDNS内のノード名ごとにではなく、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。今後、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名前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は 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。
- 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、TLS移行プロセス中に変更される可能性があるため、不適切な方法です。移行が TLS-preferred に変更されたら、 を使用して describe-replication-groupプライマリエンドポイントとリーダーエンドポイントエンドポイントをクエリAPIし、この時点から取得した DNS を使用します。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内のノードごとのDNS名前の代わりに、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、自動フェイルオーバーが有効になっているときに ElastiCache サービスによって自動的に管理されるスケーリング、パッチ適用、フェイルオーバー、およびその他の手順は、代わりにユーザーによって実行されます。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。ノードごとの古いDNS名前は削除され、クラスターを no-TLSpreferred から TLS-preferred に移行すると新しい名前が生成されるため、ノードごとのDNS名前を直接使用しないでください。これは、クライアントが TLS移行中にクラスターに接続できるように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際にレプリカ間でリードを均等に分散し、クラスターからレプリカを追加または削除するときに DNS-records を追跡するメリットもあります。
- クラスター設定エンドポイントをアプリケーションでハードコードすることは、TLS移行プロセス中に変更される可能性があるため、悪い方法です。

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あてはまります。クラスターが TLS-preferred への移行を完了した場合にのみ、 と TCPTLS接続の両方を受け入れて処理できます。したがって、転送中の暗号化が完了するまで、クラスターTLSへの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)。

- 暗号化が完了したときに SNSサービスを使用して通知を受け取る
- 暗号化が完了するとイベントを出力`describe-events`APIする の使用
- ElastiCache コンソールで暗号化が完了したことを示すメッセージを表示する

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
- をクエリして、クラスターが True `transit_encryption_enabled`に設定されていることを主張します`describe-replication-group`API。

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TLSへの接続を開き、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 情報コマンドを使用して、Valkey または Redis OSS エンジンへのより明確な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
```

Openssl (Memcached) を使用した転送中の暗号化で有効化されたノードへの接続

転送中の暗号化が有効になっている ElastiCache (Memcached) ノードからデータにアクセスするには、Secure Socket Layer () を使用するクライアントを使用する必要がありますSSL。Amazon Linux や Amazon Linux 2 で、Openssl s_client を使用することもできます。

Openssl s_client を使用して、Amazon Linux 2 または Amazon Linux で送信中の暗号化を有効にした Memcached クラスターに接続するには:

```
/usr/bin/openssl s_client -connect memcached-node-endpoint:memcached-port
```

Java を使用した TLS Memcached クライアントの作成

TLS モードでクライアントを作成するには、以下を実行して、適切な を使用してクライアントを初期化しますSSLContext。

```
import java.security.KeyStore;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManagerFactory;
import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.MemcachedClient;
public class TLSDemo {
    public static void main(String[] args) throws Exception {
        ConnectionFactoryBuilder connectionFactoryBuilder = new
 ConnectionFactoryBuilder();
        // Build SSLContext
```

```
        TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init((KeyStore) null);
        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
        // Create the client in TLS mode
        connectionFactoryBuilder.setSSLContext(sslContext);
        MemcachedClient client = new MemcachedClient(connectionFactoryBuilder.build(),
 AddrUtil.getAddresses("mycluster.fnjyzo.cfg.use1.cache.amazonaws.com:11211"));

        // Store a data item for an hour.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

を使用した TLS Memcached クライアントの作成 PHP

TLS モードでクライアントを作成するには、以下を実行して、適切な を使用してクライアントを初期化しますSSLContext。

```
<?php

/**
 * Sample PHP code to show how to create a TLS Memcached client. In this example we
 * will use the Amazon ElastiCache Auto Descovery feature, but TLS can also be
 * used with a Static mode client.
 * See Using the ElastiCache Cluster Client for PHP (https://docs.aws.amazon.com/
AmazonElastiCache/latest/dg/AutoDiscovery.Using.ModifyApp.PHP.html) for more
 information
 * about Auto Discovery and persistent-id.
 */

/* Configuration endpoint to use to initialize memcached client.
 * this is only an example */
$server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

/* Port for connecting to the cluster.
 * This is only an example     */
$server_port = 11211;

/* Initialize a persistent Memcached client and configure it with the Dynamic client
 mode  */
$tls_client =  new Memcached('persistent-id');
```

```
$tls_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);

/* Add the memcached's cluster server/s */
$tls_client->addServer($server_endpoint, $server_port);

/* Configure the client to use TLS */
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* Set your TLS context configurations values.
 * See MemcachedTLSContextConfig in memcached-api.php for all configurations */
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.mycluster.fnjyzo.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;

/* Use the created TLS context configuration object to create OpenSSL's SSL_CTX and set
 it to your client.
 * Note:  These TLS context configurations will be applied to all the servers connected
 to this client. */
$tls_client->createAndSetTLSContext((array)$tls_config);

/* test the TLS connection with set-get scenario: */

 /* store the data for 60 seconds in the cluster.
 * The client will decide which cache host will store this item.
 */
if($tls_client->set('key', 'value', 60)) {
    print "Successfully stored key\n";
} else {
    echo "Failed to set key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* retrieve the key */
if ($tls_client->get('key') === 'value') {
    print "Successfully retrieved key\n";
} else {
    echo "Failed to get key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
```

PHP クライアントの使用の詳細については、「」を参照してくださいElastiCache Cluster Client for PHP のインストール。

の保管時の暗号化 ElastiCache

データを安全に維持するために、Amazon ElastiCache と Amazon S3 では、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 VPCs と ElastiCache 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 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セキュリティを向上させる機能です。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に保存されたデータ SSDs (ソリッドステートドライブ) は常に暗号化されます。

ElastiCache は、保管時のデフォルトの (サービスマネージド) 暗号化と、AWS キー管理サービス (KMS) で独自の対称カスタマーマネージド AWS KMSキーを使用する機能を提供します。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 (サービスマネージド) 暗号化は、 GovCloud （ 米国) リージョンで使用できる唯一のオプションです。

> **Important**
>
> 既存のセルフ設計の Valkey または Redis OSS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、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
- からのカスタマーマネージドキーの使用 AWS KMS
- 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

保管 ElastiCache 時の暗号化の実装を計画するときは、保管時の ElastiCache 暗号化に関する以下の制約に注意してください。

- 保管時の暗号化は、Valkey 7.2 以降を実行するレプリケーショングループ、および Redis OSS バージョン (にスケジュールされた 3.2.6EOL、Redis OSSバージョンの有効期限スケジュール を参照)、4.0.10 以降でサポートされています。
- 保管時の暗号化は、Amazon 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 VPC。
- 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 ファミリーのノードタイプを使用している場合、 に保存されているデータは、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暗号化SSDされます。

- 保管時の暗号化にカスタマーマネージドキーを使用するオプションは、 AWS GovCloud （ us-gov-east-1 および us-gov-west-1) リージョンでは使用できません。
- 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、 に保存されているデータは、選択したカスタマーマネージド AWS KMSキー (または AWS GovCloud リージョンのサービスマネージド暗号化) でSSD暗号化されます。

- Memcached では、保管時の暗号化はサーバーレスキャッシュでのみサポートされます。
- Memcached を使用する場合、保管時の暗号化にカスタマーマネージドキーを使用するオプションはGovCloud 、 (us-gov-east-1 および us-gov-west-1) リージョンで AWS は使用できません。

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。

からのカスタマーマネージドキーの使用 AWS KMS

ElastiCache は、保管時の暗号化のために対称カスタマーマネージド AWS KMSキー (KMS キー) をサポートします。カスタマーマネージドKMSキーは、 AWS アカウントで作成、所有、管理する暗号化キーです。詳細については、AWS KMS「 Key AWS Management Service デベロッパーガイド」の「 キー」を参照してください。キーを で使用する前に、 で AWS KMSキーを作成する必要があります ElastiCache。

ルートキーを作成する AWS KMS方法については、「 Key Management Service デベロッパーガイドhttps://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html」の「キーの作成」を参照してください。 AWS

ElastiCache では、 と統合できます AWS KMS。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 と ElastiCache の統合を有効にするためのカスタマーアクションは必要ありません AWS KMS。

`kms:ViaService` 条件キーは、 AWS KMSキー (KMS キー) の使用を、指定された AWS サービスからのリクエストに制限します。`kms:ViaService` で を使用するには ElastiCache、条件キー値 `elasticache.AWS_region.amazonaws.com`と の両方 ViaService の名前を含めますdax.AWS_region.amazonaws.com。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 がユーザーに代わって ElastiCache 送信 AWS Key Management Service するリクエストを追跡できます。カスタマーマネージドキー AWS Key

Management Service に関連する へのすべてのAPI呼び出しには、対応する CloudTrail ログがあります。また、 が ElastiCache 作成した許可は、 ListGrantsKMSAPIコールを呼び出すことで確認できます。

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、レプリケーショングループに関連付けられたKMSキーを使用するようにデフォルトで暗号化されます。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、カスタマーマネージドKMSキーを使用して暗号化するか、KMS キーを使用してデフォルトの暗号化で設定された別の S3 バケットにファイルをコピーするか、ファイル自体の暗号化オプションを変更することができます。
- 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、Amazon S3 に保存されているバックアップファイルは、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、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、ハードウェア障害後に変更または復元することはできません。 AWS KMS は、少なくとも 7 日間の待機期間後にのみルートキーを削除します。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。
- 自動キーローテーションではルートキーの AWS KMSプロパティが保持されるため、ローテーションは ElastiCache データにアクセスする機能には影響しません。暗号化された Amazon ElastiCache キャッシュは、手動キーローテーションをサポートしていません。これには、新しいルートキーの作成と古いキーへの参照の更新が含まれます。詳細については、AWS 「 Key Management Service デベロッパーガイド」の「キーのローテーション AWS KMS」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。
- 権限と制限の詳細については AWS KMS、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。

ElastiCache キャッシュの作成時に保管時の暗号化を有効にすることができます。これを行うには AWS Management Console、、 AWS CLI、または を使用します ElastiCache API。

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。
- カスタマーマネージドキー – このオプションを使用すると、保管時の暗号化のために の AWS KMSキー ID/ARN を指定できます。

ルートキーの作成 AWS KMS方法については、「 キー管理サービスデベロッパーガイド」の「キーの作成」を参照してください。 AWS

目次

既存のセルフ設計の Valkey または Redis OSS クラスターでの保管時の暗号化の有効化

Valkey または Redis OSSレプリケーショングループを作成する場合にのみ、保管時の暗号化を有効にできます。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、でのクラスターの削除 ElastiCache 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。

を使用した保管時の暗号化の有効化 AWS Management Console

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KMSキーを使用してデータを暗号化します。独自の AWS KMS キーを選択するには、次の選択を行います。

- [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。

手順については step-by-step、以下を参照してください。

- Valkey (クラスターモードが無効) クラスターの作成 (コンソール)
- Valkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール)

を使用した保管時の暗号化の有効化 AWS CLI

を使用して Valkey または Redis OSS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 AWS CLI、レプリケーショングループを作成するときに --at-rest-encryption-enabled パラメータを使用します。

Valkey または Redis OSS (クラスターモードが無効) クラスターでの保管時の暗号化の有効化 (CLI)

次のオペレーションでは、プライマリレプリカと 2 つのリードレプリカである 3 つのノード OSS (--num-cache-clusters) `my-classic-rg`を持つ Valkey 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を作成します。このレプリケーショングループ (--at-rest-encryption-enabled) では保管時の暗号化が有効になっています。

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`**— `valkey`または である必要がありま`redis`す。
- **`--engine-version`**— エンジンが Redis の場合OSS、 は 3.2.6、4.0.10 以降である必要があります。
- **`--at-rest-encryption-enabled`** — 保管時の暗号化に必要です。

Example 1: レプリカを使用した Valkey または Redis OSS (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 \
    --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。

- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する (AWS CLI)
- create-replication-group

Valkey または Redis のクラスターでの保管時の暗号化の有効化 OSS (クラスターモードが有効) (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 OSS (--num-node-groups) my-clustered-rgを持つ Valkey 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作成します。各ノードには、プライマリレプリカと 2 つのリードレプリカ (--replicas-per-node-group) の 3 つのノードがあります。このレプリケーショングループ (--at-rest-encryption-enabled) では保管時の暗号化が有効になっています。

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**— valkeyまたは redisである必要がありますredis。
- **--engine-version**— エンジンが Redis の場合OSS、 は 4.0.10 以降である必要があります。
- **--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: Valkey または Redis OSS (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。

- Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループをゼロから作成する (AWS CLI)
- create-replication-group

以下の資料も参照してください。

- Amazon VPCs と ElastiCache セキュリティ
- Amazon の Identity and Access Management ElastiCache

認証と認可

AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。ElastiCache は、 IAM と Valkey および Redis OSS AUTH コマンドを使用したユーザーの認証、およびロールベースのアクセスコントロール () を使用したユーザーオペレーションの承認をサポートしますRBAC。

トピック

- ロールベースのアクセスコントロール (RBAC)
- Valkey および Redis OSS AUTH コマンドによる認証
- Valkey または Redis OSSキャッシュでの ElastiCacheアクセスコントロールの無効化

ロールベースのアクセスコントロール (RBAC)

で説明されているように、Valkey および Redis OSS AUTH コマンドでユーザーを認証する代わりにValkey および Redis OSS AUTH コマンドによる認証、Valkey 7.2 以降および Redis 6.0 以降では、ロールベースのアクセスコントロール (RBAC) OSS 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方法でもあります。

トークンが認証された場合OSSAUTHにすべての認証済みクライアントがフルキャッシュアクセスを持つ Valkey や Redis とは異なり、 RBAC ではユーザーグループを介してキャッシュアクセスを制御できます。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。

ではRBAC、以下に示すように、アクセス文字列を使用してユーザーを作成し、特定のアクセス許可を割り当てます。特定のロール (管理者、人事) と連携したユーザーグループにユーザーを割り当て、そのユーザーグループを 1 つ以上の ElastiCache キャッシュにデプロイします。これにより、同じ Valkey または Redis OSSキャッシュを使用してクライアント間のセキュリティ境界を確立し、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。

> **Note**
>
> Valkey クラスターRBACで を使用する場合、ユーザーとユーザーグループにエンジン「redis」を割り当てる必要があります。

RBAC は、Redis 6 ACLでの OSS の導入をサポートするように設計されています。 ElastiCache Valkey または Redis OSSキャッシュRBACで を使用する場合、いくつかの制限があります。

- アクセス文字列にパスワードを指定することはできません。パスワードは CreateUserまたは ModifyUser呼び出しで設定します。
- 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にアクセスコントロールリスト (ACLs) を使用する場合は、コマンドの元の名前、つまり名前を変更す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 パラメータでパスワードを指定し、 ElastiCache (Redis OSS) 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 OSS は ACL LIST コマンドを導入します。このコマンドは、各ユーザーに適用されるACLルールとともに、ユーザーのリストを返します。 ElastiCache （ Redis OSS) は `ACL LIST` コマンドをサポートしますが、Redis と同様にパスワードハッシュのサポートは含まれませんOSS。ElastiCache (Redis OSS) では、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を含む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。

  Valkey および Redis ACL USERS ElastiCache でサポートされているその他の読み取り専用コマンドOSSにはACLWHOAMI、、、および ACL CATが含まれます。 ElastiCache Valkey および Redis OSSでは、他の書き込みベースのACLコマンドはサポートされていません。
- 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

リソース	最大許容数
ユーザー数	1,000
ユーザーグループの数	100

ElastiCache (Redis OSS) RBACでの 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
- Valkey または Redis ElastiCache を使用した のキャッシュRBACへの適用 OSS
- から への移行 AUTH RBAC
- から への移行 RBAC AUTH
- ユーザーのパスワードの自動ローテーション
- IAM による認証

アクセス文字列を使用したアクセス許可の指定

ElastiCache (Redis OSS) キャッシュに対するアクセス許可を指定するには、 AWS CLI または を使用してアクセス文字列を作成し、ユーザーに割り当てます AWS Management Console。

アクセス文字列は、ユーザーに適用されるスペース区切りルールのリストとして定義されます。それらは、ユーザーが実行できるコマンドと、ユーザーが操作できるキーを定義します。コマンドを実行するには、ユーザーは、実行されているコマンドと、そのコマンドによってアクセスされているすべてのキーにアクセスできる必要があります。ルールは左から右に累積的に適用され、提供された文字列に冗長性がある場合は、提供された文字列の代わりに、より単純な文字列を使用できます。

ACL ルールの構文については、「」を参照してくださいACL。

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 に制限されます。*#### 1*.

+@category — ユーザーのアクセスは、コマンドのカテゴリーに制限されます。

アクセス文字列をユーザーに割り当てる方法については、「コンソールと を使用したユーザーとユーザーグループの作成 CLI」を参照してください。

既存のワークロードを に移行する場合は ElastiCache、ユーザーとパスワードハッシュを除きACL LIST、 を呼び出すことでアクセス文字列を取得できます。

Redis OSS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ネルに与えられます。

Redis OSS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、ACL「主要なアクセス許可」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レクターを作成します。セレ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レ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「セレクタ」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レクターをすべて削除します。

Valkey または Redis ElastiCache を使用した のキャッシュRBACへの適用 OSS

Valkey または Redis OSS ElastiCache で使用するにはRBAC、次の手順を実行します。

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 を使用したユーザーとユーザーグループの作成 CLI
- コンソールと を使用したユーザーグループの管理 CLI
- 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 を使用したユーザーとユーザーグループの作成 CLI

RBAC ユーザーのユーザー情報は、ユーザー ID、ユーザー名、およびオプションで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 とユーザー名を使用してデフォルトのユーザーを自動的に設定`"default"`し、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 OSSバージョンのデフォルトの動作と互換性があり、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があります。

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

を使用してデフォルトユーザーを変更するには AWS CLI

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-str0ng-pa))word" \
    --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-str0ng-pa))word" ^
 --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、 RBAC を ElastiCache (Redis) に使用するときは、以下のユーザーパスワードの制約に注意してくださいOSS。

- 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: , "" / @。

コンソールと を使用したユーザーの管理 CLI

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

を使用してユーザーを変更するには CLI

- `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に設定して、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。

を使用してユーザーを削除するには CLI

- 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
```

コンソールと を使用したユーザーグループの管理 CLI

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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. Amazon ElastiCache ダッシュボードで、ユーザーグループ管理 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。

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 CLI

- 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
```

を使用して新しいユーザーを追加したり、現在のメンバーを削除したりしてユーザーグループを変更するには CLI

- 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-removere 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。

を使用してユーザーグループを削除するには CLI

- 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します。

```
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、ユーザーを追加したら、実装の最終ステップRBACは、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ることです。

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

を使用してサーバーレスキャッシュにユーザーグループを追加するには AWS Management Console、以下を実行します。

- クラスターモードが無効の場合は、「Valkey (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、「Valkey または Redis OSS (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 CLI

次の AWS CLI オペレーションでは、 値 を持つ user-group-idパラメータを使用してサーバーレスキャッシュを作成します*my-user-group-id*。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`** – は `valkey`または である必要があります`redis`。
- **`--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"
```

次の AWS CLI オペレーションでは、 値 を持つ user-group-idパラメータを使用してサーバーレスキャッシュを変更します*my-user-group-id*。

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、ユーザーを追加したら、実装の最終ステップRBAC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ることです。

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

を使用してレプリケーションにユーザーグループを追加するには AWS Management Console、以下を実行します。

- クラスターモードが無効の場合は、「Valkey (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、「Valkey または Redis OSS (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。

を使用してレプリケーショングループにユーザーグループを割り当てる AWS CLI

次の AWS CLI オペレーションでは、転送中の暗号化 (TLS) が有効になっているレプリケーショングループと、 値 を持つ user-group-idsパラメータを作成します*my-user-group-id*。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** – は valkeyまたは である必要がありますredis。
- **--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"
```

次の AWS CLI オペレーションでは、転送中の暗号化 (TLS) が有効になっているレプリケーショングループと、 値 を持つ user-group-idsパラメータを変更しますmy-user-group-id。

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。

から への移行 AUTH RBAC

の説明AUTHに従って を使用してValkey および Redis OSS AUTH コマンドによる認証いて、 を使用して に移行する場合はRBAC、次の手順を使用します。

コンソールRBACを使用して から AUTH に移行するには、次の手順に従います。

コンソールRBACを使用して Valkey または Redis から OSS 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、変更するキャッシュがある 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。

Valkey または Redis から OSS AUTH RBACの使用に移行するには、次の手順を使用しますCLI。

RBAC を使用して から AUTH に移行するには CLI

- 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
```

から への移行 RBAC AUTH

を使用してRBACいて Redis に移行する場合は、OSSAUTH「」を参照してくださいから への移行 RBAC AUTH。

> **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、 を通じてそれを行う必要があります AWS CLI。詳細については、「the section called “Valkey または Redis OSSキャッシュでの ElastiCacheアクセスコントロールの無効化”」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

を使用すると AWS Secrets Manager、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager へのAPI呼び出しに置き換えて、プログラムでシークレットを取得できます。シークレットはそこに存在しないため、これは、あなたのコードを調べている誰かがシークレットを漏らさないようにするのに役立ちます。また、指定したスケジュールに従って自動的にシークレットを更新するように Secrets Manager を設定することができます。これにより、長期のシークレットを短期のシークレットに置き換えることが可能となり、侵害されるリスクが大幅に減少します。

Secrets Manager を使用すると、Secrets Manager が提供する AWS Lambda 関数を使用して、ElastiCache (Redis OSS) 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。

の詳細については AWS Secrets Manager、「 とは」を参照してください AWS Secrets Manager。

がシークレット ElastiCache を使用する方法

Valkey 7.2 には、Redis 7.0 OSS と同等の機能セットがあります。Redis OSS6 では、Valkey または Redis OSSクラスターロールベースのアクセスコントロール (RBAC) を保護するために ElastiCache 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。ではRBAC、顧客がパスワードを持つユーザーを作成する間、パスワードの値はプレーンテキストで手動で入力する必要があり、オペレータに表示されます。

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 ユーザーが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 ユーザーが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 シークレットへの参照は側面から行われません。

```
{
    "password": "strongpassword",
    "username": "user1",
    "user_arn": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1" //this is the
 bond between the secret and the user
}
```

Lambda ローテーション関数

Secrets Manager 自動パスワードローテーションを有効にするには、変更ユーザーとやり取りAPI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**
>
> 一部の AWS サービスでは、混乱した代理シナリオを避けるために、 AWS は aws:SourceArnと aws:SourceAccount グローバル条件キーの両方を使用することをお勧めします。ただし、ローテーション関数ポリシーに aws:SourceArn条件を含める場合、ローテーション関数は、その によって指定されたシークレットをローテーションするためにのみ使用できますARN。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。

問題が発生した場合は、「AWS Secrets Manager 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 Secrets Manager 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. にサインイン AWS Management Console し、 で Lambda コンソールを開きます。 https://console.aws.amazon.com/lambda/

   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] 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] を選択します。

   g. [Permissions] (許可) を選択します。

   h. 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:xxxxxxxxxx918: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:xxxxxxxxxxx: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. を使用して AWS Management Console、コンソールを使用して AWS Secrets Manager 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. の使用については AWS CLI、「 AWS Secrets Manager を使用するための自動ローテーションの設定」を参照してください。 AWS Command Line Interface

IAM による認証

トピック

- 概要
- 制限事項

- セットアップ
- 接続中

概要

IAM 認証では、キャッシュが Valkey または Redis OSSバージョン 7 以降を使用するように設定されている場合、ID OSSを使用して AWS IAM Valkey または Redis ElastiCache との への接続を認証できます。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。IAM 認証を使用して、最小権限のアクセス許可の原則に従って、個々の ElastiCache キャッシュと ElastiCache ユーザーごとにきめ細かなアクセスコントロールを設定することもできます。IAM Valkey または Redis ElastiCache を使用した の認証は、Valkey または Redis OSSAUTHまたは HELL0 コマンドで、有効期間の長い ElastiCache ユーザーパスワードの代わりに、有効期間の短いIAM認証トークンを提供することでOSS機能します。IAM 認証トークンの詳細については、AWS 「 全般のリファレンスガイド」の署名バージョン 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 ID とそれに関連するポリシーを使用して、Valkey または Redis OSS アクセスをさらに制限できます。また、フェデレーティッド ID プロバイダーから Valkey または Redis OSSキャッシュに直接アクセス許可を付与することもできます。

で を使用するには AWS IAM ElastiCache、まず認証モードを に設定して ElastiCache ユーザーを作成する必要がありますIAM。その後、IAMアイデンティティを作成または再利用できます。IAM ID には、 ElastiCache キャッシュと ElastiCache ユーザーに`elasticache:Connect`アクションを付与するための関連ポリシーが必要です。設定したら、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、キャッシュに接続するときに、Valkey または Redis OSS Client でパスワードとして有効期間の短いIAM認証トークンを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する Valkey または Redis OSS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 はIAM、 対応 ElastiCache ユーザーの接続リクエストに対してIAM認証を実行し、 との接続リクエストを検証しますIAM。

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。

- IAM 認証は、Valkey 7.2 以降および Redis OSSバージョン 7.0 以降 ElastiCache で を使用する場合に使用できます。
- 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。長期接続の場合は、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 Valkey または Redis OSSクライアントを使用することをお勧めします。
- Valkey または Redis ElastiCache との へのIAM認証された接続OSS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、次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:EpochTime、、 aws:ResourceTag/%s (関連するサーバーレスキャッシュとユーザーから) aws:SourceVpce aws:CurrentTime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。

セットアップ

IAM 認証を設定するには：

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
```

2. 以下に示すように、アカウントが新しいロールを引き受けることができるIAM信頼ポリシードキュメントを作成します。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新しい 対応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

まず、AWS SigV4 署名付きリクエスト を使用して、有効期間の短いIAM認証トークンを生成する必要があります。その後、次の例に示すように、Valkey または Redis OSS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 OSS).
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);
```

```
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct Redis OSS 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 OSS 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、認証情報プロバイダー ElastiCache を使用して でIAM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 OSS).
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);

// Create a Redis OSS credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userId, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis OSS URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce Redis OSS client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

以下は、 を認証情報プロバイダーIAMAuthTokenRequestにラップして、必要に応じて一時的な認証情報を自動的に生成する Lettuce Redis OSS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 = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
 Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
 TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userId, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
}
```

Valkey および Redis OSS AUTH コマンドによる認証

> **Note**
>
> 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は、認証RBACに を使用する必要があります。

Valkey と Redis のOSS認証トークンまたはパスワードにより、Valkey と Redis はクライアントOSS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。AUTH は、独自設計のクラスターでのみ使用できます。

トピック

- Valkey と Redis ElastiCache を使用した AUTH の の概要 OSS
- Valkey または Redis OSSクラスター ElastiCache を使用した への認証の適用
- 既存のクラスターのAUTHトークンの変更
- から への移行 RBAC AUTH

Valkey と Redis ElastiCache を使用した AUTH の の概要 OSS

Valkey または Redis OSSクラスターAUTHで ElastiCache で を使用する場合、いくつかの改良点があります。

特に、 を使用する場合は、以下のAUTHトークンまたはパスワードの制約に注意してください AUTH。

- 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、Valkey または Redis OSSクラスター ElastiCache で有効になっている転送中の暗号化に対して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、少なくとも次の 3 つの文字タイプ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、ディクショナリ単語またはわずかに変更されたディクショナリ単語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。

Valkey または Redis OSSクラスター ElastiCache を使用した への認証の適用

トークンで保護された Valkey または Redis OSSサーバーにトークン (パスワード) を入力するようにユーザーに要求できます。これを行うには、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、正しいトークンに 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、転送中の暗号化 (TLS) とAUTHトークン を有効にした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`** – `valkey`または `redis` である必要があります。
- **`--engine-version`** — エンジンが Redis の場合OSS、 は 3.2.6、4.0.10 以降である必要があります。
- **`--transit-encryption-enabled`** – 認証とHIPAA適格性に必要です。
- **`--auth-token`** – HIPAA資格を得るために必要です。この値は、このトークンで保護された Valkey または Redis OSSサーバーの正しいトークンである必要があります。
- **`--cache-subnet-group`** – HIPAA資格を得るために必要です。

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、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 Valkey 7.2 以降または Redis 5.0.6 以降であれば、この変更を行うことができます。また、転送中の暗号化が有効になってい ElastiCache る必要があります。

認証トークンを変更すると、 ROTATEと の 2 つの戦略がサポートされますSET。このROTATE戦略では、前のAUTHトークンを保持しながら、サーバーに追加トークンを追加します。この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 トークンのローテーション

Valkey または Redis OSSサーバーを新しいAUTHトークン で更新するには、 を新しいAUTHトークンとして `--auth-token`パラメータ`ModifyReplicationGroup`APIで呼び出し、 を値 `--auth-token-update-strategy`で呼び出しますROTATE。ROTATE 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、前のAUTHトークンをサポートします。AUTH トークンローテーションの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 を使用してトークンを必須AUTHに更新AUTH トークンの設定するには、「」を参照してくださいSET。

> **Note**
>
> 以前にAUTHトークンを設定していない場合、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AUTHトークンに加えて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新のAUTHトークンを最大 2 つまでサポートできます。

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ら、AUTHトークンローテーションSET戦略 (次のセクションで説明) を使用して、新しいトークンの使用を排他的に開始できます。

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

Valkey または Redis OSSサーバーを更新して 1 つの必須AUTHトークンをサポートするには、最後のAUTHトークンと同じ値を持つ `--auth-token`パラメータで `ModifyReplicationGroup`APIオペレーションを呼び出し、 値を持つ `--auth-token-update-strategy`パラメータを呼び出しますSET。SET 戦略は、以前にROTATE戦略を使用した 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、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
```

既存のクラスターでの認証の有効化

既存の Valkey または Redis OSSサーバーで認証を有効にするには、ModifyReplicationGroupAPIオペレーションを呼び出します。--auth-token パラメータModifyReplicationGroupを新しいトークンとして を呼び出し、値 --auth-token-update-strategyを持つ を呼び出しますROTATE。

ROTATE 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。AUTH トークンOSSを使用して Valkey または Redis に認証するようにすべてのクライアントアプリケーションが更新されたら、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) が有効になっている Valkey サーバーと Redis OSSサーバーでのみサポートされます。

から への移行 RBAC AUTH

の説明に従って Valkey または Redis OSS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ロールベースのアクセスコントロール (RBAC)、 に移行する場合はAUTH、次の手順を使用します。コンソールまたは を使用して移行できますCLI。

コンソールAUTHを使用して RBAC から 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、Valkey のAUTHデフォルトユーザーアクセスまたは Redis の OSSAUTHデフォルトユーザーアクセス を選択します。

7. Valkey AUTHトークンまたは Redis OSSAUTH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。

AUTHを使用して から RBAC 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、Valkey または Redis OSS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、認証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します。

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AUTHトークンを使用して、Valkey または Redis OSS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 として表示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、「Valkey および Redis OSS 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 “Valkey または Redis OSSキャッシュでの ElastiCacheアクセスコントロールの無効化”。

Valkey または Redis OSSキャッシュでの ElastiCacheアクセスコントロールの無効化

Valkey または Redis OSS TLSが有効なキャッシュのアクセスコントロールを無効にするには、次の手順に従います。キャッシュには、AUTHデフォルトのユーザーアクセスまたはユーザーグループアクセスコントロールリスト () の 2 つの異なるタイプの設定のいずれかがありますRBAC。キャッシュが AUTH設定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、キャッシュを RBAC設定に変更する必要があります。キャッシュが RBAC設定で作成された場合は、そのまま無効化できます。

で設定された Valkey または Redis OSSサーバーレスキャッシュを無効にするには RBAC

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 トークンで設定された で Valkey または Redis OSSキャッシュを無効にするには

1. AUTH トークンを に変更RBAC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っており、ユーザーグループが追加されていることを確認します。

```
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
```

で設定された Valkey または Redis OSSクラスターを無効にするには RBAC

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。

```
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 は、キャッシュデータを保護して不正アクセスから保護するために、次の手法を使用します。

- Amazon VPCs と ElastiCache 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。

トピック

Amazon VPCs と ElastiCache セキュリティ

データセキュリティは重要であるため、 ElastiCache はデータにアクセスできるユーザーを制御する手段を提供します。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異なります。

> **Important**
>
> ElastiCache クラスターの起動に Amazon EC2-Classic を使用することは廃止されました。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よく似た仮想ネットワークを定義します。Amazon を設定するVPCときは、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、セキュリティ設定の設定を行うことができます。仮想ネットワークにキャッシュクラスターを追加し、Amazon VPC 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することもできます。

このセクションでは、Amazon で ElastiCache クラスターを手動で設定する方法について説明しますVPC。この情報は、 ElastiCache と Amazon のVPC連携方法をより深く理解したいユーザーを対象としています。

トピック

- ElastiCache と Amazon について VPCs
- Amazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC
- Virtual Private Cloud の作成 (VPC)
- Amazon で実行されているキャッシュへの接続 VPC

ElastiCache と Amazon について VPCs

ElastiCache は、Amazon Virtual Private Cloud (Amazon) と完全に統合されていますVPC。ElastiCache ユーザーの場合、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 でクラスターを起動しますVPC。
- を初めて使用する場合 AWS、クラスターは Amazon にデプロイされますVPC。デフォルトが自動的にVPC作成されます。
- デフォルトがありVPC、クラスターの起動時にサブネットを指定しない場合、クラスターはデフォルトの Amazon で起動しますVPC。

詳細については、「サポートされているプラットフォームの検出」と「デフォルトの があるかどうかVPC」を参照してください。

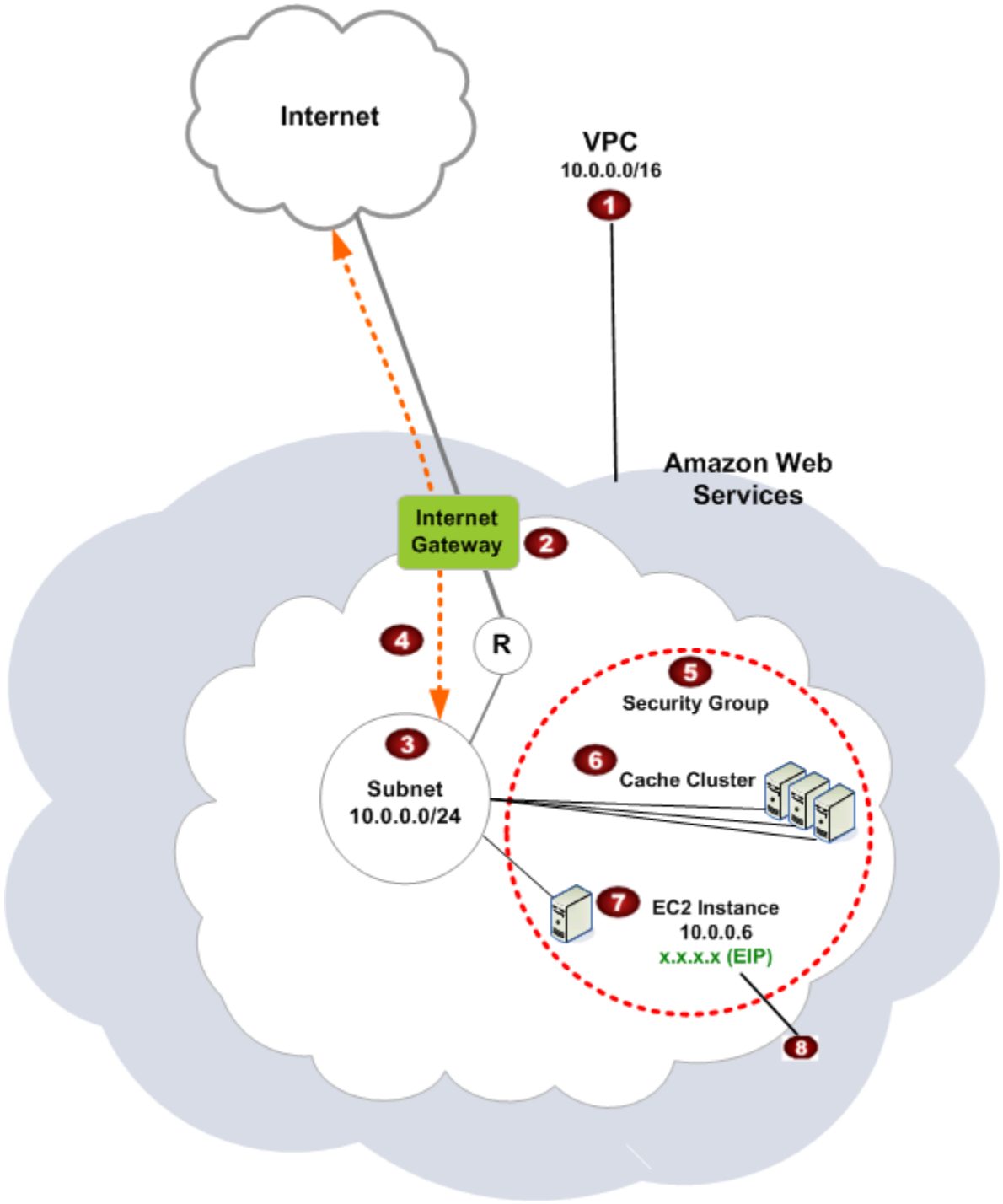
Amazon Virtual Private Cloud を使用すると、従来のデータセンターによく似た仮想ネットワークを AWS クラウドに作成できます。IP アドレス範囲の選択VPC、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティ設定の設定など、Amazon を設定できます。

の基本的な機能は仮想プライベートクラウドでも同じ ElastiCache です。クラスターが Amazon の内外にデプロイされているかどうかにかかわらず、ソフトウェアのアップグレード、パッチ適用、障害検出、復旧 ElastiCache を管理しますVPC。

ElastiCache Amazon の外部にデプロイされたキャッシュノードVPCには、エンドポイント/DNS 名前が解決される IP アドレスが割り当てられます。これにより、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスからの接続が提供されます。Amazon VPCプライベートサブネットで ElastiCache クラスターを起動すると、すべてのキャッシュノードに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 ElastiCache での の概要 VPC

次の図と表は、Amazon VPC環境と、Amazon で起動される ElastiCache クラスターと Amazon EC2 インスタンスを示していますVPC。



1	Amazon VPCは、独自の IP アドレスブロックが割り当てられている AWS クラウドの分離された部分です。
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、Amazon Simple Storage Service (Amazon S3) など、Amazon の の外部で実行されている他の AWS リソースへのアクセスを提供しますVPC。
3	Amazon VPCサブネットは Amazon の IP アドレス範囲のセグメントVPCで、セキュリティと運用上のニーズに合わせて AWS リソースを分離できます。
4	Amazon 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間のネットワークトラフィックをVPC指示します。Amazon VPCには暗黙的なルーターがあり、この図では R の円で表されています。
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トラフィックとアウトバウンドトラフィックを制御します。
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インスタンスには、サブネットのアドレス範囲からのプライベート IP アドレスがあります。Amazon EC2インスタンスは、同じサブネット内の任意のキャッシュノードに接続できます。
8	Amazon EC2インスタンスがインターネットからアクセスできるようにするVPCには、Elastic IP アドレスと呼ばれる静的なパブリックアドレスをインスタンスに割り当てる必要があります。

前提条件

Amazon 内に ElastiCache クラスターを作成するにはVPC、Amazon が以下の要件を満たしているVPC必要があります。

- Amazon は、非専用 Amazon EC2インスタンスを許可VPCする必要があります。専用インスタンステナンシー用にVPC設定された Amazon ElastiCache では を使用できません。

- Amazon VPC. ElastiCache uses がサブネットグループをキャッシュして、そのサブネット内のサブネットと IP アドレスを選択して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には、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。
- CIDR 各サブネットの ブロックは、メンテナンスアクティビティ ElastiCache で使用するスペア IP アドレスを提供するのに十分な大きさである必要があります。

ルーティングとセキュリティ

Amazon でルーティングを設定VPCして、トラフィックフロー (インターネットゲートウェイや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を使用すると、Amazon VPCは Amazon で実行されていない他の AWS リソースに直接アクセスできますVPC。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持つことを選択した場合は、 経由でインターネットバウンドトラフィックをルーティングVPN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と、追加の帯域幅料金が発生します。

Amazon VPC セキュリティグループを使用して、Amazon 内の ElastiCache クラスターと Amazon EC2インスタンスを保護することができますVPC。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**
>
> 基盤となる IP アドレスは変更される可能性があるため、DNS名前を使用してキャッシュノードに接続することを強くお勧めします。

Amazon VPCドキュメント

Amazon VPCには、Amazon を作成して使用する方法を説明する独自のドキュメントセットがありますVPC。次の表は、Amazon VPCガイドへのリンクを示しています。

説明	ドキュメント
Amazon の使用を開始する方法 VPC	Amazon の開始方法 VPC
VPC を使用して Amazon を使用する方法 AWS Management Console	Amazon VPC ユーザーガイド

説明	ドキュメント
すべての Amazon VPC コマンドの完全な説明	Amazon EC2 コマンドラインリファレンス (Amazon VPC コマンドは Amazon EC2リファレンスにあります)
Amazon VPCAPIオペレーション、データ型、エラーの完全な説明	Amazon EC2 API リファレンス (Amazon VPCAPIオペレーションは Amazon EC2リファレンスにあります)
オプションのIPsecVPN接続の最後に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に関する情報	とは AWS Site-to-Site VPN

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 で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC

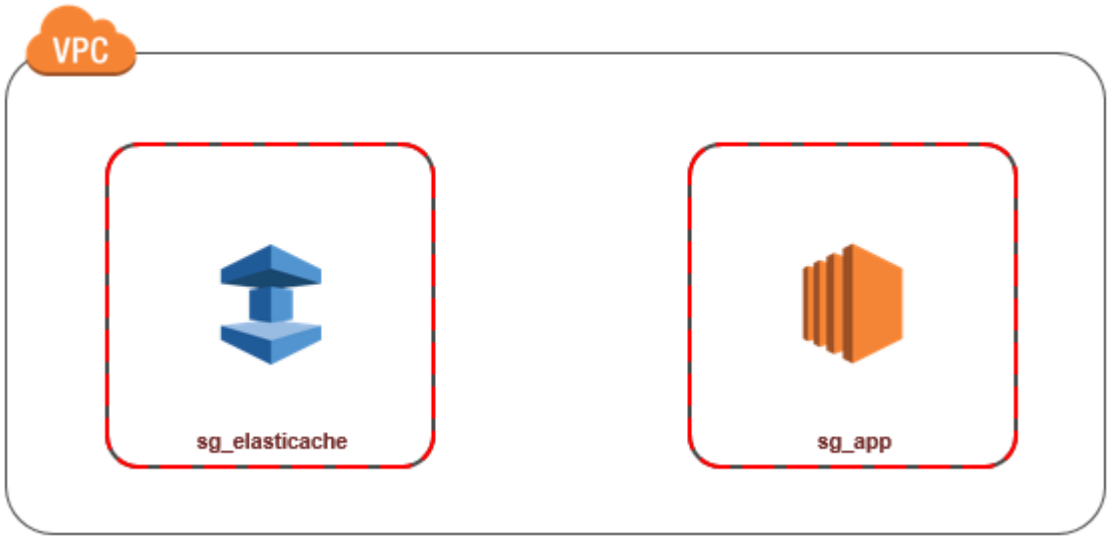
Amazon は、Amazon 内のキャッシュにアクセスするための以下のシナリオ ElastiCache をサポートしていますVPC。

目次

ElastiCache キャッシュと Amazon EC2 インスタンスが同じ Amazon にある場合のキャッシュへのアクセス VPC

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 のキャッシュに接続する必要がある場合ですVPC。

このシナリオを以下に図表で示します。



同じ 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最も簡単な方法はVPC、以下を実行することです。

1. キャッシュVPC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、このセキュリティグループのカスタムルールを作成して、キャッシュの作成時にキャッシュに割り当てたポートと、キャッシュTCPへのアクセスに使用する IP アドレスを使用してアクセスを許可できます。

   Memcached キャッシュのデフォルトのポートは 11211 です。

   Valkey キャッシュと Redis OSSキャッシュのデフォルトポートは です6379。
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) VPC 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、このセキュリティグループのルールを設定して、ポート 22 経由でEC2インスタンスTCPへのアクセスを許可できます。
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カスタムルールをキャッシュのセキュリティグループに作成します。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はサブネットを他のアベイラビリティーゾーンの任意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。

別のVPCセキュリティグループからの接続を許可するルールをセキュリティグループに作成するには

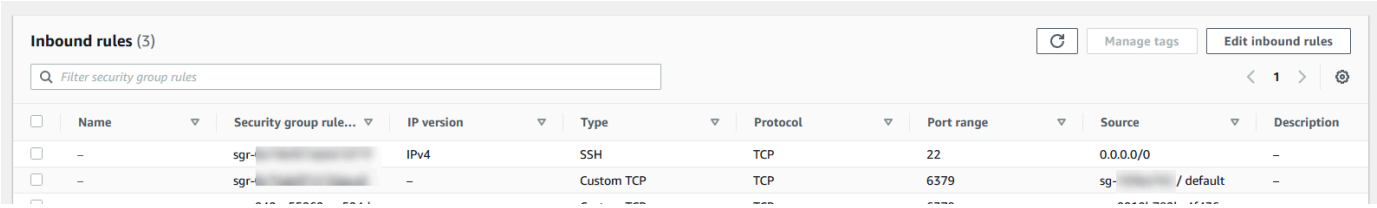
1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/vpc で Amazon VPCコンソールを開きます。
2. ナビゲーションペインで、[Security Groups] を選択します。
3. キャッシュに使用するセキュリティグループを選択または作成します。インバウンドルールで、インバウンドルールの編集 を選択し、ルールの追加 を選択します。このセキュリティグループは、他のセキュリティグループのメンバーへのアクセスを許可します。
4. タイプ からカスタムTCPルール を選択します。

   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。

      Memcached キャッシュのデフォルトのポートは 11211 です。

      Valkey および Redis OSSキャッシュとレプリケーショングループのデフォルトポートは です6379。

   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。



ElastiCache キャッシュと Amazon EC2 インスタンスが異なる Amazon にある場合のキャッシュへのアクセス VPCs

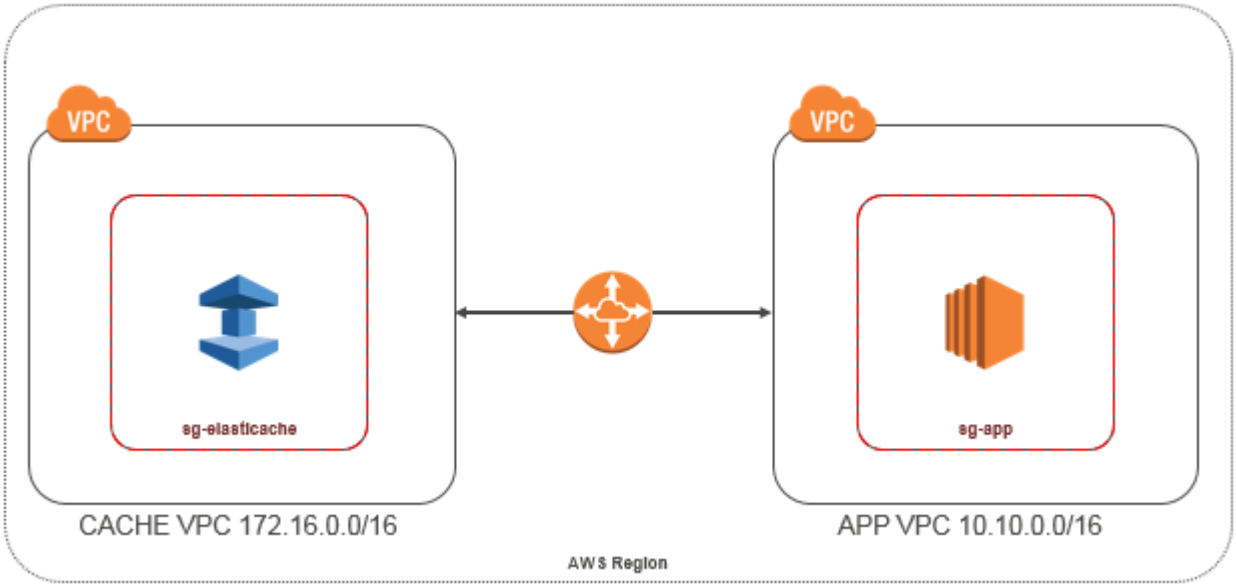
キャッシュがアクセスに使用するEC2インスタンスVPCとは異なる にある場合、キャッシュにアクセスする方法はいくつかあります。キャッシュとEC2インスタンスが異なるVPCsが、同じリージョンにある場合は、VPCピアリングを使用できます。キャッシュとEC2インスタンスが異なるリージョンにある場合は、リージョン間のVPN接続を作成できます。

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VPCsの異なる Amazon 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VPCsの異なる Amazon 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VPCsの異なる Amazon にある場合のキャッシュへのアクセス

次の図は、Amazon VPCピアリング接続を使用して、同じリージョンEC2内の別の Amazon VPC インスタンスからキャッシュにアクセスする方法を示しています。



同じリージョンVPC内の別の Amazon の 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 ピアリング接続は、プライベート IP アドレスを使用してトラフィックをルーティングVPCsできる 2 つの間のネットワーク接続です。どちらのインスタンスも、同じネットワーク内にあるかのように相互に通信VPCできます。独自の Amazon 間VPCs、または 1 つのリージョン内の別の AWS アカウントの Amazon VPC との間でVPCピアリング接続を作成できます。Amazon VPCピアリングの詳細については、VPCドキュメント を参照してください。

> **Note**
>
> DNS に適用された設定によってはVPCs、ピアリングされた の名前解決が失敗することがあります ElastiCache VPC。これを解決するには、DNSホスト名とDNS解決の両方を有効にするVPCs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VPCを介して別の Amazon のキャッシュにアクセスするには

1. 2 つの IP 範囲が重複VPCsしていないことを確認してください。重複していないと、ピアリングできません。
2. 2 つの をピアリングしますVPCs。詳細については、「Amazon VPC ピアリング接続の作成と承認」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「ピアリング接続のルートテーブルの更新」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
172.16.0.0/16	local
10.10.0.0/16	pcx-a894f1c1

Destination	Target
10.10.0.0/16	local
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. ピアリングされた 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するように、 ElastiCache キャッシュのセキュリティグループを変更しますVPC。詳細については、「参照ピアVPCセキュリティグループ」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 リージョンで VPCs と VPN の接続をアタッチし、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 アカウントと共有した後、アカウント所有者はトランジットゲートウェイVPCsにそれらをアタッチできます。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、マルチキャストトラフィックを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、ドメインに関連付けるVPC添付ファイル経由で送信できるようにするトランジットゲートウェイのマルチキャストドメインを作成できます。

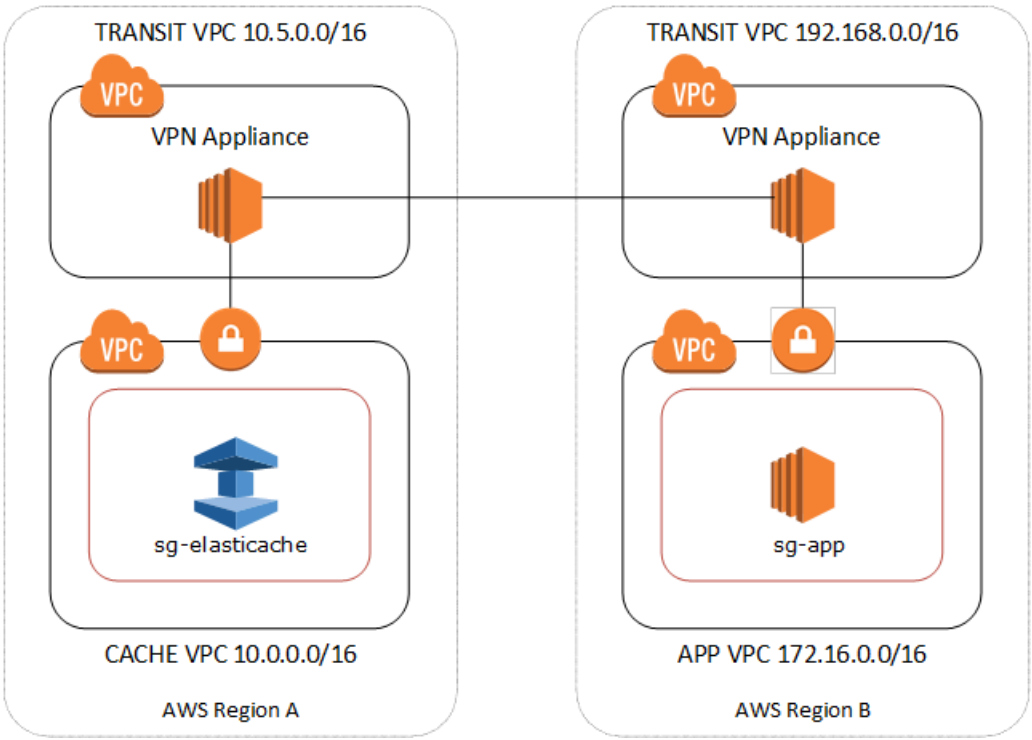
また、異なる AWS リージョンのトランジットゲートウェイ間にピアリング接続アタッチメントを作成することもできます。これにより、異なるリージョン間でトランジット・ゲートウェイのアタッチメント間でトラフィックをルーティングできます。

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VPCsの異なる Amazon にある場合のキャッシュへのアクセス

Transit の使用 VPC

VPC ピアリングを使用する代わりに、地理的に分散VPCsした複数のリモートネットワークを接続するためのもう 1 つの一般的な戦略は、グローバルネットワークトランジットセンターとして機能するトランジットを作成するVPCことです。トランジットは、ネットワーク管理VPCを簡素化し、複数のVPCsリモートネットワークを接続するために必要な接続の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VPCs間での接続

Transit Amazon 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VPCは、別のリージョンVPCの「スポーク」の ElastiCache キャッシュに接続できます。

別の AWS リージョンVPC内の別の 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 トランジット・ゲートウェイ」を参照してください。
2. アプリケーションとキャッシュのVPCルーティングテーブルを更新VPCsして、 VGW (仮想プライベートゲートウェイ) と VPNアプライアンスを介し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) による動的ルーティングの場合、ルートは自動的に伝播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

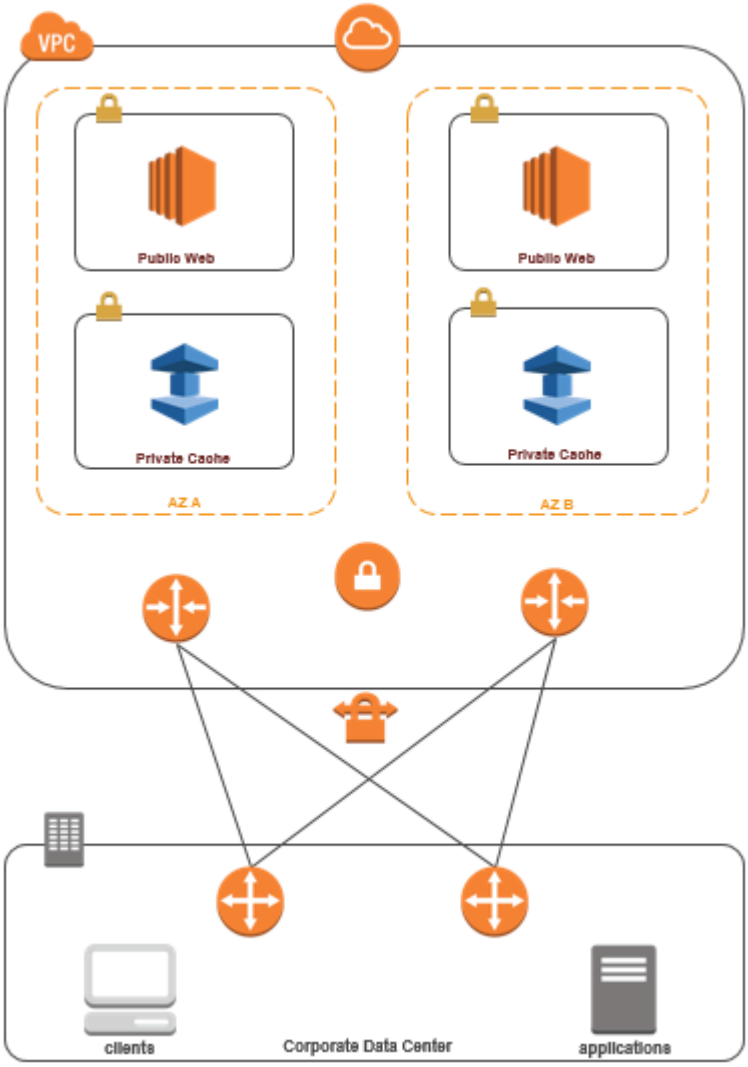
もう 1 つの考えられるシナリオは、顧客のデータセンター内のクライアントまたはアプリケーションが の ElastiCache キャッシュにアクセスする必要があるハイブリッドアーキテクチャですVPC。このシナリオは、 VPN または Direct Connect を介して、顧客とVPCデータセンター間の接続がある場合にもサポートされます。

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。



経由でデータセンター ElastiCache から に接続する VPN

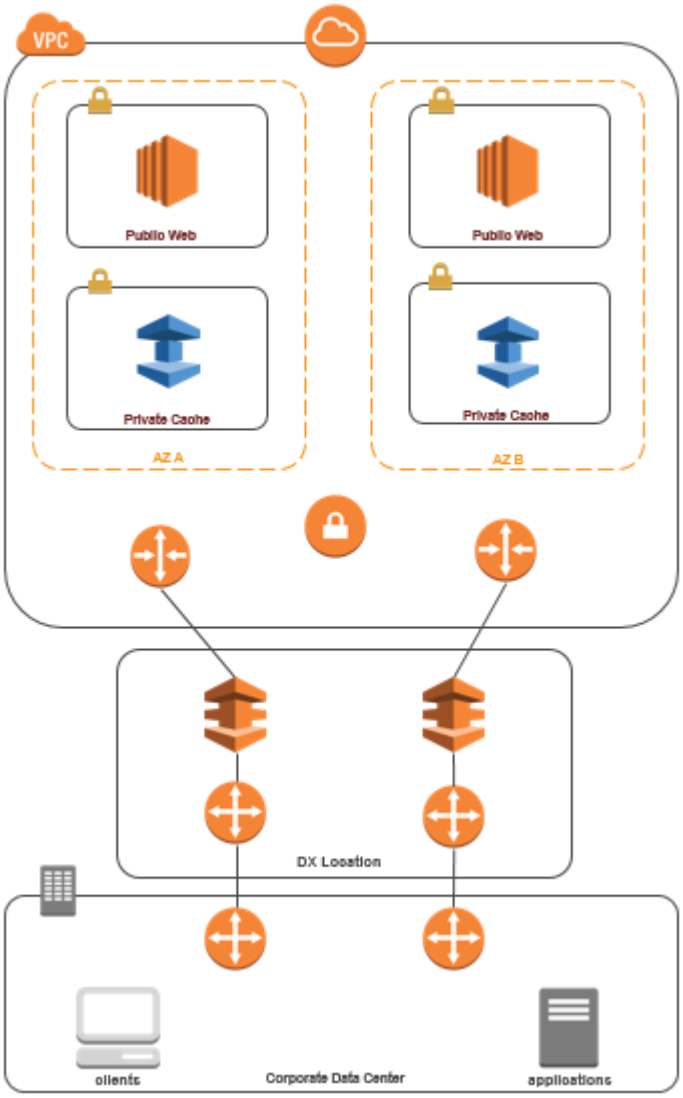
VPN 接続を介してオンプレミスアプリケーションVPCから のキャッシュにアクセスするには

1. ハードウェア Virtual Private Gateway を に追加してVPN接続を確立しますVPC。詳細については、「 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VPC」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 ルートとの動的ルーティングの場合、自動的に伝播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、ネットワークレイテンシーと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 の作成 (VPC)

この例では、アベイラビリティーゾーンごとにプライベートサブネットVPCを持つ Amazon を作成します。

Amazon の作成 VPC (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、 で Amazon VPCコンソールを開きますhttps://console.aws.amazon.com/vpc/。
2. VPC ダッシュボードで、「 の作成VPC」を選択します。
3. 作成、選択、VPCその他の のリソース。
4. アベイラビリティーゾーンの数 (AZs) で、サブネットを起動するアベイラビリティーゾーンの数を選択します。
5. パブリックサブネットの数 で、 に追加するパブリックサブネットの数を選択しますVPC。
6. プライベートサブネットの数 で、 に追加するプライベートサブネットの数を選択しますVPC。

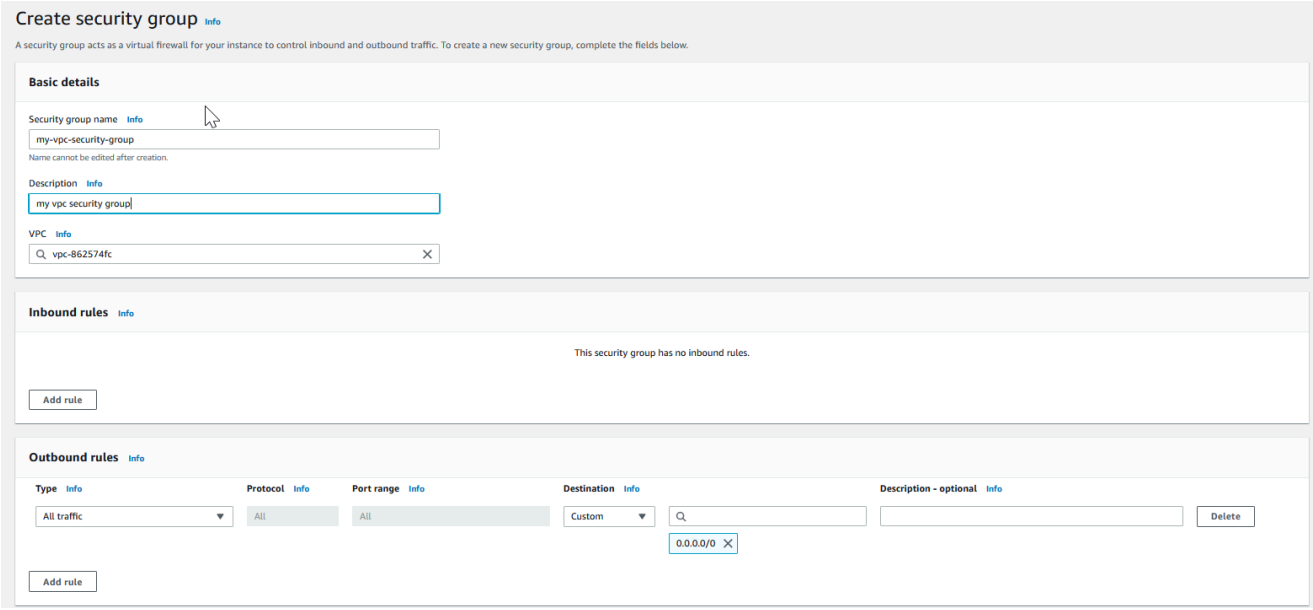
   > **Tip**
   >
   > サブネットの識別子と、どちらがパブリックで、どちらがプライベートであるかを書き留めておきます。この情報は、クラスターを起動し、Amazon EC2インスタンスを Amazon に追加するときに後で必要になりますVPC。

7. Amazon VPC セキュリティグループを作成します。このグループは、キャッシュクラスターと Amazon EC2インスタンスに使用します。

   a. Amazon VPC Management コンソールのナビゲーションペインで、セキュリティグループを選択します。

   b. [Create Security Group (セキュリティグループの作成)] を選択します。

   c. 対応するボックスにセキュリティグループの名前と説明を入力します。VPC ボックスで、Amazon の識別子を選択しますVPC。



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。このルールでは、Secure Shell () を使用して Amazon EC2インスタンスに接続できますSSH。

   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。新しいルールの作成ボックスで、 を選択しSSH、ルールの追加 を選択します。

   d. HTTP アクセスを許可するには、新しいインバウンドルールに次の値を設定します。

      - タイプ: 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、Amazon でキャッシュクラスターを起動する準備が整いましたVPC。

- サブネットグループの作成
- Memcached クラスター (CLI) の作成 (コンソール).
- Valkey (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 で実行されているキャッシュへの接続 VPC

この例では、Amazon で Amazon EC2インスタンスを起動する方法を示していますVPC。その後、このインスタンスにログインし、Amazon で実行されている ElastiCache キャッシュにアクセスできますVPC。

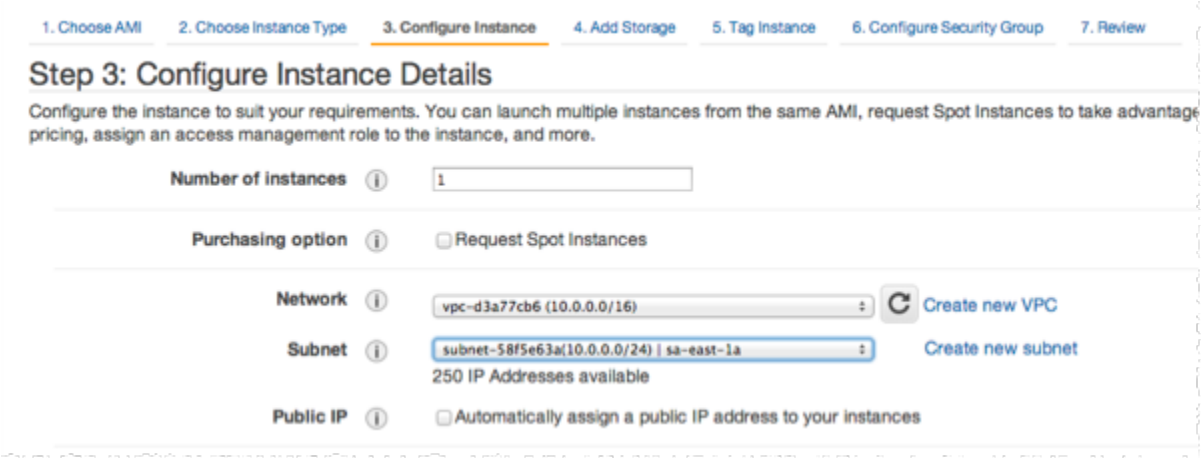
Amazon VPC (コンソール) で実行されているキャッシュへの接続

この例では、Amazon インスタンスを Amazon EC2に作成しますVPC。この Amazon EC2インスタンスを使用して、Amazon で実行されているキャッシュノードに接続できますVPC。

> **Note**
>
> Amazon の使用の詳細についてはEC2、「Amazon ドキュメント」の「Amazon EC2 入門ガイド」を参照してください。 EC2

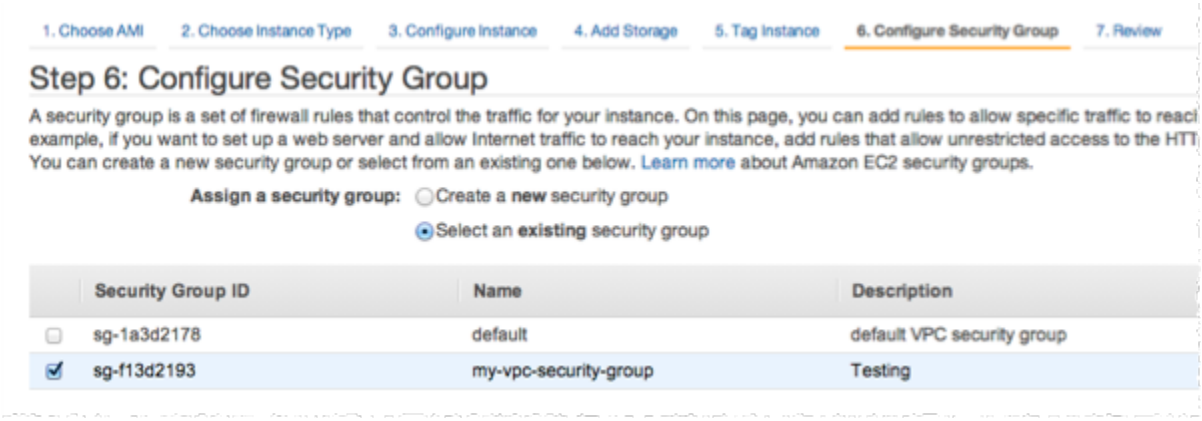
Amazon EC2コンソールVPCを使用して Amazon で Amazon EC2インスタンスを作成するには

1. にサインイン AWS Management Console し、 で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. コンソールで、[インスタンスを起動] をクリックし、以下の手順を実行します。
3. Amazon マシンイメージの選択 (AMI) ページで、64 ビットの Amazon Linux を選択し AMI、「 を選択」を選択します。
4. [インスタンスタイプの選択] ページで、[3. インスタンスの設定] を選択します。
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 を選択しますVPC。
   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。



すべての設定が正しいことを確認したら、[4. ストレージの追加] を選択します。

6. [ストレージの追加] ページで、[5. インスタンスのタグ付け] を選択します。
7. タグインスタンスページで、Amazon EC2インスタンスの名前を入力し、6 を選択します。セキュリティグループの設定] を選択します。
8. [セキュリティグループの設定] ページで [既存のセキュリティグループを選択する] 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 インスタンスの Amazon EC2 セキュリティグループ」を参照してください。



Amazon VPC セキュリティグループの名前を選択し、レビューと起動 を選択します。

9. [インスタンス作成の確認] 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ら、起動 を選択します。

これで、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。Amazon EC2インスタンスに接続するには、この IP アドレスを使用する必要があります。

Elastic IP アドレスを割り当てるには (コンソール)

1. で Amazon VPCコンソールを開きますhttps://console.aws.amazon.com/vpc/。
2. ナビゲーションリストで、Elastic IPsを選択します。
3. [Elastic IP アドレスを割り当てる] をクリックします。
4. [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。

   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**
>
> Amazon EC2インスタンスからまだ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整いました。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

- 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」と入力します。

```
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!
```

Memcached または Redis VPCを使用して に接続できるようになりました。

Memcached VPCを使用した への接続

1. で ElastiCache コンソールに移動https://console.aws.amazon.com/elasticache/し、キャッシュクラスター内のノードの 1 つのエンドポイントを取得します。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。

2. telnet を使用して、ポート 11211 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 11211
```

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。

```
add mykey 0 3600 28
This is the value for mykey
```

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。

```
VALUE mykey 0 28
This is the value for my key
END
```

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
```

Redis VPCを使用した への接続

1. で ElastiCache コンソールに移動https://console.aws.amazon.com/elasticache/し、キャッシュクラスター内のノードの 1 つ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。

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. で ElastiCache コンソールに移動https://console.aws.amazon.com/elasticache/し、キャッシュクラスター内のノードの 1 つ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OSS。
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 アカウントに追加料金が発生しないように、これらの例を試した後、不要になったリソースは必ず削除 AWS してください。

ElastiCache API およびインターフェイスVPCエンドポイント (AWS PrivateLink)

インターフェイスエンドポイント を作成することで、 VPCと Amazon ElastiCache API VPCエンドポイント間のプライベート接続を確立できます。インターフェイスエンドポイントは を搭載しています[AWS 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、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。

内のインスタンスは、Amazon ElastiCache API エンドポイントと通信するためにパブリック IP アドレスを必要としVPCません。また、インスタンスは、使用可能な ElastiCache APIオペレーションを使用するためにパブリック IP アドレスを必要としません。VPC と Amazon 間のトラフィック ElastiCache 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 ユーザーガイド」の「Elastic Network Interface」を参照してください。EC2

- VPC エンドポイントの詳細については、「Amazon 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。 VPC
- オペレーションの詳細については ElastiCache API、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、Amazon VPCは次の形式で使用できるDNSエンドポイント名を提供します。

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

詳細については、「Amazon 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。 は、 内のすべてのAPIアクションへの呼び出し ElastiCache をサポートしますVPC。 VPC

> **Note**
>
> プライベートDNSホスト名は、 内の 1 つのVPCエンドポイントでのみ有効にできます VPC。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 VPCエンドポイントのインターフェイス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、 VPCを使用して から利用できます AWS PrivateLink。

VPC エンドポイントポリシーはエンドポイントで ElastiCache APIサポートされています。デフォルトでは、 エンドポイントを介してオペレーションへの ElastiCache API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「エンドポイントを使用したサービスへのアクセスの制御」を参照してください。

のインターフェイスVPCエンドポイントの作成 ElastiCache API

Amazon ElastiCache API VPCコンソールまたは を使用して、Amazon のVPCエンドポイントを作成できます AWS CLI。詳細については、「Amazon 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。 VPC

インターフェイスVPCエンドポイントを作成したら、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 リージョンでは、 `elasticache.cn-north-1.amazonaws.com.cn` を北京に、を寧夏`elasticache.cn-northwest-1.amazonaws.com.cn`に使用し、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 のVPCエンドポイントポリシーの作成 ElastiCache API

へのアクセスを制御するエンドポイントポリシーをVPCエンドポイントにアタッチできます ElastiCache API。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「エンドポイントを使用したサービスへのアクセスの制御」を参照してください。

Example VPC Valkey または Redis を使用したアクションの ElastiCache APIエンドポイントポリシー OSS

以下は、 のエンドポイントポリシーの例です ElastiCache API。エンドポイントにアタッチされると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リスト ElastiCache API された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 VPC ElastiCache (Memcached) APIアクションのエンドポイントポリシー

以下は、 のエンドポイントポリシーの例です ElastiCache API。エンドポイントにアタッチされると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リスト ElastiCache API された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"
  ],
  "Resource": "*"
 }]
}
```

Example VPC 指定された AWS アカウントからのすべてのアクセスを拒否する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 アカウントを拒否します *123456789012* エンドポイントを使用したリソースへのすべてのアクセス。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 () 環境で実行されている独自設計のクラスターに指定できるサブネット (通常はプライベートVPC) のコレクションです。

Amazon で独自設計のクラスターを作成する場合はVPC、サブネットグループを使用する必要があります。ElastiCache は、そのサブネットグループを使用して、ノードに関連付けるサブネット内のサブネットと IP アドレスを選択します。

ElastiCache はデフォルトのサブネットグループを提供するか、新しいIPv4サブネットグループの作成を選択できます。ではIPv6、 IPv6 CIDR ブロックを使用して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、 IPv6または のいずれかの Discovery IP タイプを選択する必要がありますIPv4。

ElastiCache Serverless はサブネットグループリソースを使用しないため、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、サブネットとサブネットグループを作成して活用し、 ElastiCache リソースへのアクセスを管理する方法について説明します。

Amazon VPC環境でのサブネットグループの使用の詳細については、「」を参照してください ElastiCache 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。

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、 内のキャッシュに指定するサブネットのコレクションです VPC。でキャッシュを起動するときはVPC、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、そのキャッシュサブネットグループ ElastiCache を使用して、そのサブネット内の 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。

ネットワークタイプIPV4として を選択すると、デフォルトのサブネットグループが使用可能になるか、新しいサブネットグループを作成するかを選択できます。 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サブネット内の IP アドレスを選択します。デュアルスタックまたは を選択した場合IPV6、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、「 でサブネットを作成するVPC」を参照してください。

次の手順では、 `mysubnetgroup` (コンソール ） 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ElastiCache API。

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、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
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] 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 を選択しますVPC。

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。Selected subnets パネルで、アベイラビリティーゾーンまたはローカルゾーンとIDsプライベートサブネットの管理と選択をクリックし、「 を選択」を選択します。

5. 表示された確認メッセージで、Close を選択します。

新しいサブネットグループは、 ElastiCache コンソールのサブネットグループリストに表示されます。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 「」トピックを参照してください。create-cache-subnet-group.

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 でキャッシュを起動できますVPC。詳細については、以下を参照してください。

- [Memcached クラスター] – Memcached クラスターを起動するには、「Memcached クラスター (CLI) の作成 (コンソール)」を参照してください。ステップ 7.a (アドバンスト Memcached 設定) で、VPCサブネットグループを選択します。
- スタンドアロン Valkey または Redis OSSクラスター – 単一ノード Valkey または Redis OSSクラスターを起動するには、「」を参照してくださいValkey (クラスターモードが無効) クラスターの作成 (コンソール) 。ステップ 7.a (高度な Redis OSS設定) で、VPCサブネットグループを選択します。
- Valkey または Redis OSS (クラスターモードが無効) レプリケーショングループ – で Valkey または Redis OSS (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VPC、「」を参照してくださいValkey または Redis OSS (クラスターモードが無効) レプリケーショングループをゼロから作成する。ステップ 7.b (高度な Redis OSS設定) で、VPCサブネットグループを選択します。
- Valkey または Redis OSS (クラスターモードが有効) レプリケーショングループ – Valkey または Redis OSS (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 (高度な Redis OSS設定) 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か、サブネットグループIDsに関連付けられたサブネットのリストを変更できます。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. 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 [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、変更するサブグループのラジオボタンを選択し、[変更] を選択します。
4. [選択されたサブネット] パネルで [管理] をクリックします。
5. 選択したサブネットに変更を加え、[選択] をクリックします。
6. [変更を保存] 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 「」トピックを参照してください。modify-cache-subnet-group.

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. 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 [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 [削除] 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)

を使用して AWS CLI、次のパラメータdelete-cache-subnet-groupを使用して コマンドを呼び出します。

- `--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 「」トピックを参照してください。delete-cache-subnet-group.

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 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management （ IAM) の使用方法は、 で行う作業によって異なります 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、管理者は必要な認証情報とアクセス許可を提供します。より多くの ElastiCache 機能を使用して作業を行う際に、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解しておくと、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、 へのフルアクセスがある可能性があります 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必要があるかを判断するのはお客様の仕事です。その後、サービスユーザーのアクセス許可を変更するリクエストをIAM管理者に送信する必要があります。このページの情報を確認して、 の基本概念を理解しますIAM。会社IAMで を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 の ElastiCache 連携方法 IAM。

IAM 管理者 – IAM管理者の場合は、 へのアクセスを管理するためのポリシーの作成方法の詳細を知りたい場合があります ElastiCache。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IAM、「」を参照してくださいAmazon のアイデンティティベースのポリシーの例 ElastiCache。

アイデンティティを使用した認証

認証は、アイデンティティ認証情報 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) する必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインすると、管理者は以前に IAMロールを使用して ID フェデレーションをセットアップしていました。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、 AWS サインイン ユーザーガイドの「 へのサインイン方法 AWS アカウント」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号化して署名します。 AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。推奨される方法を使用してリクエストに署名する方法の詳細については、 IAM ユーザーガイドの「リクエストの署名 AWS API」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、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、AWS IAM Identity Center 「 ユーザーガイド」の「多要素認証の使用」および「 ユーザーガイド」の「多要素認証の使用 (MFA ） AWS」を参照してください。 IAM

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての および リソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインイン ID 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してアクセスします。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー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「ルートユーザーの認証情報を必要とするタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、 では、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アクセスするために ID プロバイダーとのフェデレーション AWS のサービス の使用を要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブアイデンティティプロバイダー、 AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを通じて提供された認証情報 AWS のサービス を使用してアクセスするすべてのユーザーのユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、それらはロールを引き受け、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したり、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 ユーザーとは、1 人のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を持つIAMユーザーを作成する代わりに、一時的な認証情報に依存することをお勧めします。ただし、IAMユーザーとの長期的な認証情報を必要とする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ユーザーのコレクションを指定する ID 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、という名前のグループがありIAMAdmins、そのグループにIAMリソースを管理するアクセス許可を付与できます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー

ザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、 IAM ユーザーガイドの (ロールではなく) IAM ユーザーを作成するタイミングを参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内の ID です。ユーザーと似ていますがIAM、特定の人物には関連付けられていません。IAM ロール を切り替える AWS Management Console ことで、 でロールを一時的に引き受けることができます。または AWS API オペレーション を AWS CLI 呼び出すか、カスタム を使用してロールを引き受けることができますURL。ロールを使用する方法の詳細については、 IAM ユーザーガイドの「ロールを引き受ける方法」を参照してください。

IAM 一時的な認証情報を持つ ロールは、次の状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、 IAM ユーザーガイドの「サードパーティー ID プロバイダーのロールの作成」を参照してください。IAM Identity Center を使用する場合は、アクセス許可セットを設定します。ID が認証された後にアクセスできる内容を制御するために、IAMIdentity Center はアクセス許可セットを のロールに関連付けますIAM。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ユーザーアクセス許可 – IAM ユーザーまたはロールは、特定のタスクに対して異なるアクセス許可を一時的に引き受けるIAMロールを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、別のアカウントの誰か (信頼できるプリンシパル) が自分のアカウントのリソースにアクセスすることを許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセス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「 のクロスアカウントリソースアクセスIAM」を参照してください。
- クロスサービスアクセス — 他の の機能 AWS のサービス を使用するものもあります AWS のサービス。例えば、 サービスで呼び出しを行うと、そのサービスが Amazon でアプリケーションを実行EC2したりAmazon S3にオブジェクトを保存したりするのが一般的です。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、を呼び出すプリンシパルのアクセス許可と 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を使用します。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権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「アクセスセッションの転送」を参照してください。
- サービスロール – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受けるIAMロールです。IAM 管理者は、 内からサービスロールを作成、変更、削除できますIAM。詳細については、IAM「 ユーザーガイド」の「 にアクセス許可を委任するロールの作成 AWS のサービス」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、サービスにリンクされたロールのアクセス許可を表示することはできますが、編集することはできません。

- Amazon で実行されているアプリケーション EC2 – 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インスタンス内にアクセスキーを保存するよりも望ましいです。 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 ロールが含まれており、EC2インスタンスで実行されているプログラムが一時的な認証情報を取得できるようにします。詳細については、IAM「 ユーザーガイド」のIAM「ロールを使用して 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アクセス許可を付与する」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「 (ユーザーではなく) IAMロールを作成するタイミング」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には AWS 、ポリシーを作成して AWS ID またはリソースにアタッチします。ポリシーは 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 にJSONドキュメント AWS として保存されます。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「ポリシーの概要」を参照してください。

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。必要な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ユーザーに付与するには、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行に使用する方法に関係なく、アクションのアクセス許可を定義します。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 からロール情報を取得できますAPI。

アイデンティティベースのポリシー

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーを作成する方法については、「 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。マネージドポリシーは、 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選択」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースにアタッチするJSONポリシードキュメントです。リソースベースのポリシーの例としては、IAMロール信頼ポリシー と Amazon S3 バケットポリシー があります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーに

よって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーIAMでは、 から AWS 管理ポリシーを使用することはできません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) 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アクセス許可を持っているかを制御します。ACLs 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されません。

Amazon S3、および Amazon VPCは AWS WAF、 をサポートするサービスの例ですACLs。の詳細についてはACLs、「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) 概要」を参照してください。

その他のポリシータイプ

AWS は、追加であまり一般的ではない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がIAMエンティティ (IAMユーザーまたはロール) に付与できる最大アクセス許可を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM「 ユーザーガイド」のIAM「エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) の最大アクセス許可を指定するJSONポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント 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 (SCPs) をアカウントのいずれかまたはすべてに適用できます。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SCPを制限します AWS アカウントのルートユーザー。Organizations と の詳細についてはSCPs、AWS Organizations 「 ユーザーガイド」の「サービスコントロールポリシー」を参照してください。

- 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、「 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。 IAM

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係している場合にリクエストを許可するかどうか AWS を決定する方法については、 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。 IAM

Amazon と の ElastiCache 連携方法 IAM

IAM を使用して へのアクセスを管理する前に ElastiCache、 で使用できるIAM機能について説明します ElastiCache。

IAM Amazon で使用できる機能 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	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	Yes
ポリシー条件キー	あり
ACLs	可能
ABAC (ポリシーのタグ)	可能
一時的な認証情報	あり

IAM 機能	ElastiCache サポート
プリンシパル権限	あり
サービスロール	あり
サービスリンクロール	可能

ElastiCache および他の AWS のサービスがほとんどのIAM機能とどのように連携するかの概要については、 IAM ユーザーガイドのAWS 「 と連携する のサービスIAM」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のポリシーのサポート: あり

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシー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 ID ベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否される条件を指定できます。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IAMJSON「ポリシー要素リファレンス」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 ID 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 のアイデンティティベースのポリシーの例 ElastiCache。

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: なし

リソースベースのポリシーは、リソースにアタッチするJSONポリシードキュメントです。リソースベースのポリシーの例としては、IAMロール信頼ポリシー と Amazon S3 バケットポリシー があります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの

場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、リソースベースのポリシーのプリンシパルとして、別のアカウントのアカウントまたはIAMエンティティ全体を指定できます。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 にある場合 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「 でのクロスアカウントリソースアクセスIAM」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のサポート: あり

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action`要素は、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を記述します。ポリシーアクションは通常、関連付けられた AWS APIオペレーションと同じ名前です。一致するAPIオペレーションがないアクセス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 によって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 ID 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 のアイデンティティベースのポリシーの例 ElastiCache。

のポリシーリソース ElastiCache

ポリシーリソースのサポート: あり

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの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 のリストを確認するにはARNs、「サービス認証リファレンス」の「Amazon によって定義されるリソース ElastiCache」を参照してください。各リソースARNの を指定できるアクションについては、「Amazon で定義されるアクション ElastiCache」を参照してください。

ElastiCache ID 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 のアイデンティティベースのポリシーの例 ElastiCache。

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: あり

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルが、どのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、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 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば、IAMユーザー名でタグ付けされている場合にのみ、リソースにアクセスする許可をIAMユーザーに付与できます。詳細については、「 ユーザーガイド」のIAM「ポリシー要素: 変数とタグ」を参照してください。 IAM

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、 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。 IAM

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 ID 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 のアイデンティティベースのポリシーの例 ElastiCache。

のアクセスコントロールリスト (ACLs) ElastiCache

をサポートACLs： はい

アクセスコントロールリスト (ACLs) 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アクセス許可を持っているかを制御します。ACLs 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されません。

を使用した属性ベースのアクセスコントロール (ABAC) ElastiCache

サポート ABAC (ポリシーのタグ): はい

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認証戦略です。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、の最初のステップですABAC。次に、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するときに、オペレーションを許可するABACポリシーを設計します。

ABAC は、急速に成長している環境で役立ち、ポリシー管理が面倒になる状況に役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

の詳細についてはABAC、 IAM ユーザーガイドの「 とはABAC」を参照してください。を設定する手順を含むチュートリアルを表示するにはABAC、 IAM ユーザーガイドの「属性ベースのアクセスコントロールを使用する (ABAC) 」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: あり

一時的な認証情報を使用してサインインすると機能 AWS のサービス しない場合があります。一時的な認証情報 AWS のサービス を使用する方法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 を使用する方法IAMを参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成

されます。ロールの切り替えの詳細については、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。

AWS CLI または を使用して、一時的な認証情報を手動で作成できます AWS API。その後、これらの一時的な認証情報を使用して にアクセスできます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「」の「一時的なセキュリティ認証情報IAM」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

転送アクセスセッションをサポート (FAS): はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可と 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を使用します。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権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「アクセスセッションの転送」を参照してください。

のサービスロール ElastiCache

サービスロールのサポート: あり

サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受けるIAMロールです。IAM 管理者は、 内からサービスロールを作成、変更、削除できますIAM。詳細については、IAM「 ユーザーガイド」の「 にアクセス許可を委任するロールの作成 AWS のサービス」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 機能が壊れる可能性があります。 ElastiCache がガイダンスを提供する場合にのみ、サービスロールを編集します。

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: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示することはできますが、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、AWS 「 と連携するサービス IAM」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 のアイデンティティベースのポリシーの例 ElastiCache

デフォルトでは、ユーザーとロールにはリソースを作成または変更 ElastiCacheするアクセス許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または を使用してタスクを実行することはできません AWS API。必要な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ユーザーに付与するには、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

これらのポリシードキュメント例を使用して IAM ID ベースのJSON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。

各リソースタイプの の形式など ElastiCache、 で定義されるアクションとARNsリソースタイプの詳細については、「サービス認証リファレンス」の「Amazon のアクション、リソース、および条件キー ElastiCache」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、誰かがアカウント内の ElastiCache リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを開始し、最小権限のアクセス許可に移行 – ユーザーとワークロードへのアクセス許可の付与を開始するには、多くの一般的なユースケースのアクセス許可を付与するAWS マネージドポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM「 ユーザーガイド」の「 AWS 管理ポリシーAWS 」または ジョブ機能の 管理ポリシーを参照してください。
- 最小権限のアクセス許可を適用する - IAMポリシーでアクセス許可を設定する場合、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用してアクセス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「 のポリシーとアクセス許可IAM」を参照してください。
- IAM ポリシーの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、アクションと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 を使用して送信する必要があることを指定できますSSL。などの特定の を通じてサービスアクションが使用されている場合 AWS のサービス、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 AWS CloudFormation。詳細については、IAM「 ユーザーガイド」のIAMJSON「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、安全で機能的なアクセス許可を確保する – IAM Access Analyzer は、ポリシーがポリシー言語 (JSON) とIAMベストプラクティスに準拠するように、新規および既存のIAMポリシーを検証します。IAM Access Analyzer には、安全で機能的なポリシーの作成に役立つ 100 を超えるポリシーチェックと実用的なレコメンデーションが用意されています。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。
- 多要素認証が必要 (MFA) – でIAMユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、 をオンにMFAしてセキュリティを強化します。API オペレーションが呼び出されるMFAタイミングを要求するには、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。

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「 のセキュリティのベストプラクティスIAM」を参照してください。 IAM

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、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 内の ElastiCache リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成

すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または のみを呼び出すユーザーに対して、最小限のコンソールアクセス許可を付与する必要はありません AWS API。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 ElastiCache 引き続きコンソールを使用できるようにするには、`ConsoleAccess` または `ReadOnly` AWS 管理ポリシーを ElastiCacheエンティティにアタッチします。詳細については、「 ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。 IAM

自分の権限の表示をユーザーに許可する

この例では、IAMユーザーがユーザー ID にアタッチされているインラインポリシーとマネージドポリシーを表示できるようにするポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI または を使用してプログラムでこのアクションを実行するアクセス許可が含まれています AWS API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
```

```
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon 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

ElastiCache 以下の情報は、 および の使用時に発生する可能性のある一般的な問題を診断して修正するのに役立ちますIAM。

トピック

- でアクションを実行する権限がありません ElastiCache
- iam を実行する権限がありません。PassRole
- AWS アカウント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ありません ElastiCache

がアクションを実行する権限がないと AWS Management Console 通知した場合は、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、 にロールを渡すことができるようにポリシーを更新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。

次のエラー例は、 という名前の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とすると発生します 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。

AWS アカウント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 (ACLs) をサポートするサービスでは、これらのポリシーを使用して、リソースへのアクセスをユーザーに許可できます。

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については、「」を参照してくださいAmazon と の ElastiCache 連携方法 IAM。
- 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。

- サードパーティー にリソースへのアクセスを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を提供する (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にロールとリソースベースのポリシーを使用する違いについては、 IAM ユーザーガイドの「 のクロスアカウントリソースアクセスIAM」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の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 リソースへのアクセス許可の管理の概要
- Amazon 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

ElastiCache リソースへの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、リソースを作成またはアクセスするためのアクセス許可はアクセス許可ポリシーによって管理されます。アカウント管理者は、アクセス許可ポリシーを ID (ユーザー、グループ、ロール) IAM にアタッチできます。さらに、Amazon はリソースへのアクセス許可ポリシーのアタッチ ElastiCache もサポートしています。

> **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、「 ユーザーガイド」のIAM「ベストプラクティス」を参照してください。 IAM

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。
- ID プロバイダーIAMを介して で管理されるユーザー :

  ID フェデレーションのロールを作成します。IAM ユーザーガイドの「サードパーティー ID プロバイダーのロールの作成 (フェデレーション) 」の指示に従います。
- IAM ユーザー :
  - ユーザーが担当できるロールを作成します。「 ユーザーガイド」のIAM「ユーザーのロールを作成する」の手順に従います。 IAM
  - (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) へのアクセス許可を追加する」の手順に従います。

トピック

- Amazon ElastiCache 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
- AWS Amazon の マネージドポリシー ElastiCache
- Amazon 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

- リソースレベルのアクセス許可
- 条件キーの使用
- Amazon のサービスリンクロールの使用 ElastiCache
- 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 リソースとオペレーション

ElastiCache リソースタイプとその のリストを確認するにはARNs、「サービス認証リファレンス」の「Amazon によって定義されるリソース ElastiCache」を参照してください。各リソースARNの を指定できるアクションについては、「Amazon で定義されるアクション ElastiCache」を参照してください。

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、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。では ElastiCache、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、そのユーザーにキャッシュクラスターを作成するアクセス許可を付与するとします。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、ユーザーが属する AWS アカウントは、キャッシュクラスターリソースを所有します。
- キャッシュクラスターを作成するアクセス許可を持つIAMロールを AWS アカウントに作成するとします。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 アカウントは、キャッシュクラスターリソースを所有します。

リソースへのアクセスの管理

アクセス権限ポリシー では、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス許可ポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、Amazon のコンテキストIAMでの の使用について説明しますElastiCache。IAM サービスに関する詳細情報は提供されません。詳細なIAMドキュメントについては、IAM「 ユーザーガイド」の「 とはIAM」を参照してください。IAM ポリシーの構文と説明については、「 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。 IAM

IAM ID にアタッチされたポリシーは、ID ベースのポリシー (IAM ポリシー) と呼ばれます。リソースに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

- ID 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

ID ベースのポリシー (IAM ポリシー)

IAM ID にポリシーをアタッチできます。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。
- アクセス許可ポリシーをロールにアタッチする (クロスアカウントアクセス許可を付与する) – 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、クロスアカウントアクセス許可を付与できます。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 (アカウント B など) 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 管理者はIAMロールを作成し、アカウント A のリソースに対するアクセス許可を付与するアクセス許可ポリシーをロールにアタッチします。
  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 管理者は、アカウント B の任意のユーザーにロールを引き受けるアクセス許可を委任できます。これにより、アカウント B のユーザーがアカウント A のリソースを作成または

アクセスできるようになります。場合によっては、ロールを引き受ける AWS サービスアクセス許可を付与することもできます。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。

を使用してアクセス許可IAMを委任する方法の詳細については、「 ユーザーガイド」の「アクセス管理」を参照してください。 IAM

以下は、ユーザーが AWS アカウントのDescribeCacheClustersアクションを実行できるようにするポリシーの例です。 は、 API アクションARNsのリソースを使用した特定のリソースの識別 ElastiCache 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
      }
   ]
}
```

でのアイデンティティベースのポリシーの使用の詳細については ElastiCache、「」を参照してくださいAmazon 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。ユーザー、グループ、ロール、および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

Amazon ElastiCache リソース (「」を参照Amazon ElastiCache リソースとオペレーション) ごとに、サービスは一連のAPIオペレーションを定義します (「アクション」を参照)。これらのAPIオペレーションのアクセス許可を付与するには、ポリシーで指定できる一連のアクション ElastiCache を定義します。例えば、クラスターリソースでは ElastiCache、、CreateCacheCluster、DeleteCacheClusterおよび のアクションが定義されますDescribeCacheCluster。API オペレーションを実行するには、複数のアクションに対するアクセス許可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 – ポリシーでは、Amazon リソースネーム (ARN) を使用して、ポリシーが適用されるリソースを識別します。詳細については、「Amazon ElastiCache 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 – アクションキーワードを使用して、許可または拒否するリソース操作を特定します。例えば、指定された に応じてEffect、アクセスelasticache:CreateCacheCluster許可は Amazon ElastiCache CreateCacheClusterオペレーションを実行するためのユーザーアクセス許可を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) では、ポリシーがアタッチされているユーザーが暗黙的なプリンシパルです。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 ポリシーの構文と説明の詳細については、「 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。 IAM

すべての Amazon ElastiCache API アクションを示す表については、「」を参照してください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。

ポリシーでの条件の指定

アクセス許可を付与する場合、IAMポリシー言語を使用して、ポリシーを有効にす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。ポリシー言語で条件を指定する方法の詳細については、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、「」を参照してください条件キーの使用。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全体のキーの完全なリストについては、 IAM ユーザーガイドの「条件に使用可能なキー」を参照してください。

AWS Amazon の マネージドポリシー ElastiCache

AWS マネージドポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。AWS マネージドポリシーは、多くの一般的なユースケースに対するアクセス許可を提供するように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

AWS マネージドポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小権限のアクセス許可を付与しない場合があることに注意してください。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS マネージドポリシーで定義されているアクセス許可は変更できません。が マネージドポリシーで AWS 定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しいAPIオペレーション AWS のサービス が既存のサービスで使用できるようになったときに AWS 、マネージドポリシーを更新する可能性が最も高いです。

詳細については、「 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。 IAM

AWS マネージドポリシー： ElastiCacheServiceRolePolicy

IAM エンティティ ElastiCacheServiceRolePolicy にアタッチすることはできません。このポリシーは、 がユーザーに代わって ElastiCache アクションを実行できるようにするサービスにリンクされたロールにアタッチされます。

このポリシーでは ElastiCache 、 がキャッシュを管理するために必要な AWS リソースをユーザーに代わって管理できます。

- `ec2` – VPCエンドポイント (サーバーレスキャッシュ用) 、Elastic Network Interfaces (ENIs) (独自設計クラスター用) 、セキュリティグループなど、キャッシュノードにアタッチするEC2ネットワークリソースを管理します。
- `cloudwatch` – サービスから にメトリクスデータを送信します CloudWatch。
- `outposts` – AWS Outposts 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 コンソールと AWS マネージドポリシーリファレンスガイド ElastiCacheServiceRolePolicy にあります。

AWS マネージドポリシー： AmazonElastiCacheFullAccess

ID IAM にAmazonElastiCacheFullAccessポリシーをアタッチできます。

このポリシーでは、プリンシパルが AWS マネジメントコンソール ElastiCache を使用して へのフルアクセスを許可します。

- elasticache — すべての にアクセスしますAPIs。
- 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EC2リソース (VPC、サブネット、セキュリティグループ) について説明し、VPCエンドポイントの作成を許可します (サーバーレスキャッシュの場合) 。
- kms — CMKsのカスタマーマネージド型の使用を許可します encryption-at-rest。
- cloudwatch — メトリクスへのアクセスを許可して、コンソールに ElastiCache メトリクスを表示します。
- 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に AWS Outposts 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のSNSトピックを入力するために使用されます。

AmazonElastiCacheFullAccess ポリシーは、 IAM コンソールと AWS マネージドポリシーリファレンスガイド AmazonElastiCacheFullAccess にあります。

AWS マネージドポリシー： AmazonElastiCacheReadOnlyAccess

ID IAM にAmazonElastiCacheReadOnlyAccessポリシーをアタッチできます。

このポリシーでは、 AWS マネジメントコンソール ElastiCache を使用してプリンシパルが への読み取り専用アクセスを許可します。

- elasticache — 読み取り専用 Describe にアクセスしますAPIs。

AmazonElastiCacheReadOnlyAccess ポリシーは、 IAM コンソールと AWS マネージドポリシーリファレンスガイド AmazonElastiCacheReadOnlyAccess にあります。

ElastiCache AWS マネージドポリシーの更新

このサービスがこれらの変更の追跡を開始 ElastiCache してからの AWS の管理ポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、 ElastiCache ドキュメント履歴ページのRSSフィードにサブスクライブします。

変更	説明	日付
AmazonElastiCacheFullAccess - 既存ポリシーへの更新	ElastiCache に、サーバーレスキャッシュの管理を許可し、コンソールを介してすべてのサービス機能の使用を可能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 – 既存ポリシーへの更新	ElastiCache サーバーレスキャッシュリソースのVPCエンドポイントの管理を許可する新しいアクセス許可を追加しました。	2023 年 11 月 27 日
ElastiCache 変更の追跡を開始しました	ElastiCache は、 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 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

このトピックでは、アカウント管理者がアクセス許可ポリシーを ID (ユーザー、グループ、ロール) IAM にアタッチできる ID ベースのポリシーの例を示します。

> **Important**
>
> まず、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を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 リソースへの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。

- AWS Amazon の マネージドポリシー ElastiCache
- カスタマーマネージドポリシーの例

Redis を使用する場合のアクセス許可ポリシーの例を次に示しますOSS。

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

Memcached を使用する場合のアクセス許可ポリシーの例を次に示します。

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

```
            "elasticache:CreateServerlessCache",
            "elasticache:CreateCacheCluster",
            "elasticache:DescribeServerlessCaches",
            "elasticache:DescribeCacheClusters",
            "elasticache:ModifyServerlessCache",
            "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`値の最後に指定されたIAMロール名に対する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。アクセス許可ポリシーをIAMロールにアタッチすると、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許可を取得します。

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を示す表については、「」を参照してください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。

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: ユーザーに通話を許可する IAM

CreateServiceLinkedRole API」を参照)。または、 ElastiCache サービスにリンクされたロールを作成する必要があります。

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許可と組み合わせると、このセクションのポリシーの例は追加のアクセス許可を付与します。例は、 AWS SDKsと にも関連しています AWS CLI。

IAM ユーザーとグループの設定手順については、 IAM ユーザーガイドの「最初のIAMユーザーと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、本番環境で使用する前に、必ず徹底的にテストしてください。シンプルに見える一部の ElastiCache アクションでは、 ElastiCache コンソールの使用時にサポートする他のアクションが必要になる場合があります。例えば、 は ElastiCache キャッシュクラスターを作成するアクセス許可`elasticache:CreateCacheCluster`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 コンソールは多数の アクションDescribeと Listアクションを使用してコンソールリストを入力します。

例

- 例 1: ElastiCache リソースへの読み取り専用アクセスをユーザーに許可する
- 例 2: ユーザーに一般的な 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: ユーザーに通話を許可する IAM CreateServiceLinkedRole API
- 例 5: IAM認証を使用してサーバーレスキャッシュへの接続をユーザーに許可する

例 1: ElastiCache リソースへの読み取り専用アクセスをユーザーに許可する

次のポリシーは、ユーザーがリソースを一覧表示できるようにするアクセス許可 ElastiCache アクションを付与します。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もできます。次のポリシーは、これらの一般的なシステム管理者タスクに対して ElastiCache アクションを実行する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": "ECAllowMutations",
      "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

次のポリシーは、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": "ECAllowAll",
      "Effect":"Allow",
```

```
      "Action":[
          "elasticache:*"
      ],
      "Resource":"*"
      }
   ]
}
```

例 4: ユーザーに通話を許可する IAM CreateServiceLinkedRole API

次のポリシーでは、ユーザーが IAM `CreateServiceLinkedRole` を呼び出すことを許可します API。このタイプのアクセス許可ポリシーは、ミュータティブ ElastiCache オペレーションを呼び出すユーザーに付与す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:AWS ServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

例 5: IAM認証を使用してサーバーレスキャッシュへの接続をユーザーに許可する

次のポリシーでは、すべてのユーザーが 2023-04-01 から までのIAM認証を使用して任意のサーバーレスキャッシュに接続できます2023-06-30。

```
{
  "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。多くの ElastiCache APIアクションは、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。すべてのIAMポリシーステートメントは、リソースに対して実行されるアクションにアクセス許可を付与します。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で、ユーザーが変更できるリソースを制限できます。リソースごとにアクセス許可を制限するには、 でリソースを指定しますARN。

ElastiCache リソースタイプとその のリストを確認するにはARNs、「サービス認証リファレンス」の「Amazon で定義されるリソース ElastiCache」を参照してください。各リソースARNの を指定できるアクションについては、「Amazon で定義されるアクション ElastiCache」を参照してください。

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定する条件を指定できます。では ElastiCache、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内のキーをポリシーで指定したキー値と比較できます。詳細については、IAMJSON「ポリシー要素: 条件」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。

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、特定のリクエストで個々のパラメータのセットを制御する条件を指定するIAMアクセス許可ポリシーを記述します。次に、 IAMコンソールを使用して作成したIAMユーザー、グループ、またはロールにポリシーを適用します。

条件を適用するには、IAMポリシーステートメントに条件情報を追加します。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。

Valkey または Redis を使用する場合のこのアクセス許可ポリシーの例を次に示しますOSS。

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

```
        }
    ]
}
```

Memcached を使用する場合のこのアクセス許可ポリシーの例を次に示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.r5.large"
                    ]
                }
            }
        }
    ]
}
```

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。

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: きめ細かなパラーメータコントロールのための IAM ポリシー条件の使用

このセクションでは、以前にリストした ElastiCache パラメータに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の例を示します。

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:M aximumECPUPerSecond : サーバーレスキャッシュの ECPU1 秒あたりの最大値を指定します。指定された条件を使用すると、顧客は 1 秒ECPUsあたりに特定の数を超え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(s) 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:CacheNodeType: Memcached を使用して、ユーザーが作成できる NodeType(s) 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"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
        }
    ]
}
```

5. 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"
                }
            }
        }
    ]
}
```

6. elasticache:ReplicasPerNodeGroup: ノードあたりのレプリカを 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"
                }
            }
        }
    ]
}
```

7. elasticache:EngineVersion: エンジンバージョン 5.0.6 の使用を指定します。

```
{
    "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:EngineVersion": "5.0.6"
                }
            }
```

```
        }
    ]
}
```

8. elasticache:EngineVersion: Memcached エンジンバージョン 1.6.6 の使用を指定する

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "1.6.6"
                }
            }
        }
    ]
}
```

9. elasticache:EngineType: Valkey または Redis OSS エンジンのみを使用して指定します。

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

10elasticache:AtRestEncryptionEnabled: 暗号化を有効にした場合にのみレプリケーショングループを作成することを指定します。

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

11Elasticache:TransitEncryptionEnabled

a. CreateReplicationGroup アクションelasticache:TransitEncryptionEnabledの条件キーfalseを に設定して、 TLS が使用され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ことを指定します。

```
{
    "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falseでelasticache:TransitEncryptionEnabled条件キーが に設定されている場合、 が使用されTLSていない (つまり、 CreateReplicationGroup に設定されたTransitEncryptionEnabledパラメータtrueまたは に設定されたTransitEncryptionModeパラメータがリクエストに含まれていない) 場合にのみ、リクエストが許可されますrequired。

b. CreateReplicationGroup アクションの elasticache:TransitEncryptionEnabled conditon 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ことを指定します。

```
{
    "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trueでelasticache:TransitEncryptionEnabled条件キーがに設定されている場合、CreateReplicationGroupリクエストに に設定されたTransitEncryptionEnabledパラメータtrueと に設定されたTransitEncryptionModeパラメータが含まれている場合にのみ、リクエストが許可されますrequired。

c. ModifyReplicationGroup アクションtrueのelasticache:TransitEncryptionEnabled を 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変更できることを指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:ModifyReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "BoolIfExists": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
```

```
        }
    ]
}
```

ModifyReplicationGroup アクションのポリシーtrueでelasticache:TransitEncryptionEnabled条件キーが に設定されている場合、ModifyReplicationGroupリクエストに に設定されたTransitEncryptionModeパラメータが含まれている場合にのみ、リクエストが許可されますrequired。に設定されたTransitEncryptionEnabledパラメータはtrueオプションで含めることもできますが、この場合は を有効にする必要はありませんTLS。

12elasticache:AutomaticFailoverEnabled: 自動フェイルオーバーが有効になっている場合にのみレプリケーショングループを作成す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"
                }
            }
        }
```

```
    ]
}
```

13elasticache:M ultiAZEnabled: マルチ AZ を無効にしてレプリケーショングループを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"
                }
            }
        }
    ]
}
```

14Elasticache:ClusterModeEnabled: レプリケーショングループを作成できるのは、クラスターモードが有効になっている場合のみであることを指定します。

```
{
```

```
    "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"
                }
            }
        }
    ]
}
```

15.elasticache:AuthTokenEnabled: レプリケーショングループを作成できるのはAUTHトークンが有効になっている場合のみであることを指定します。

```
{
    "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": {
                "Bool": {
                    "elasticache:AuthTokenEnabled": "true"
                }
            }
        }
    ]
}
```

16.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"
                }
            }
        }
    ]
}
```

17.elasticache:KmsKeyId: カスタマーマネージド AWS KMSキーの使用を指定します。

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

18. elasticache:CacheParameterGroupName: クラスター上の組織から特定のパラメータを持つ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"
                }
            }
        }
    ]
}
```

19. elasticache:CacheParameterGroupName: Memcached を使用して、クラスター上の組織からの特定のパラメータを持つ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「」のみの使用を制限する例です。

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

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
```

```
    ]
}
```

20elasticache:CreateCacheCluster: リクエストタグProjectが欠落しているかDev、 QAまたは と等しくない場合、CreateCacheClusterアクションを拒否しますProd。

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

21. elasticache:CacheNodeType: cacheNodeType cache.r5.large または cache.r6g.4xlarge とタグ CreateCacheCluster での許可Project=XYZ。

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

22elasticache:CacheNodeType: cacheNodeType cache.r5.large または cache.r6g.4xlarge とタグ CreateCacheCluster での許可Project=XYZ。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "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を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 のサービスリンクロールの使用 ElastiCache

Amazon ElastiCache は AWS Identity and Access Management （ IAM) サービスにリンクされたロール を使用します。サービスにリンクされたロールは、Amazon などの AWS サービスに直接リンクされる一意のタイプのIAMロールです ElastiCache。Amazon ElastiCache サービスにリンクされたロールは、Amazon によって事前定義されています ElastiCache。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と、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon のセットアップ ElastiCache が簡単になります。ロールは AWS アカウント内に既に存在しますが、Amazon ElastiCache ユースケースにリンクされており、事前定義されたアクセス許可があります。Amazon のみがこれらのロールを ElastiCache 引き受け、これらのロールのみが事前定義されたアクセス許可ポリシーを使用できます。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 ElastiCache 、リソースにアクセスするために必要なアクセス許可を誤って削除できないため、Amazon リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスの詳細については、「 AWS と連携するサービスIAM」を参照し、「サービスにリンクされたロール」列で「はい」があるサービスを探します。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

目次

- 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 のサービスにリンクされたロールの説明の編集 ElastiCache
  - 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 のサービスにリンクされたロールの削除 ElastiCache
  - 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 のサービスにリンクされたロールのアクセス許可 ElastiCache

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 サービスにリンクされたロールを作成できるようにするには

次のポリシーステートメントをその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/AWS
 ServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "elasticache.amazonaws.com"}}
}
```

IAMエンティティが AWS ServiceRoleForElastiCache サービスにリンクされたロールを削除できるようにするには

次のポリシーステートメントをその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/AWS
 ServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "elasticache.amazonaws.com"}}
}
```

または、 AWS マネージドポリシーを使用して Amazon へのフルアクセスを提供することもできます ElastiCache。

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は、 IAM コンソール、CLI、または を使用して作成できますAPI。

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. にサインイン AWS Management Console し、 でIAMコンソールを開きますhttps://console.aws.amazon.com/iam/。
2. IAM コンソールのナビゲーションペインで、ロール を選択します。次に、新しいロールの作成を選択します。
3. 信頼されたエンティティの種類を選択 の下で、AWS Service (サービス) を選択します。
4. または で、サービスを選択してそのユースケースを表示しますElastiCache。
5. [Next: Permissions] (次へ: 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であることに注意してください。次: タグ を選択します。
7. タグは、サービスにリンクされたロールではサポートされないことに注意してください。次: レビュー を選択します。
8. 「オプショナル」ロールの説明 で、サービスにリンクされた新しいロールの説明を編集します。

9. ロール情報を確認し、ロールの作成 を選択します。

サービスにリンクされたロールの作成 (IAM CLI)

からIAMオペレーションを使用して AWS Command Line Interface 、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (CLI)

次のオペレーションを使用してください。

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

サービスにリンクされたロールの作成 (IAM API)

を使用してIAMAPI、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (API)

を使用する CreateServiceLinkedRole API を呼び出します。リクエストで、サービス名`elasticache.amazonaws.com`を指定します。

Amazon のサービスにリンクされたロールの説明の編集 ElastiCache

Amazon ElastiCache では、 AWS ServiceRoleForElastiCache 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した後は、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、 を使用してロールの説明を編集できますIAM。

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)

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の説明を編集できます。

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)

からIAMオペレーションを使用して AWS Command Line Interface 、サービスにリンクされたロールの説明を編集できます。

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) ロールの現在の説明を表示するには、 IAMオペレーション AWS CLI に を使用しますget-role。

   Example

   ```
   $ aws iam get-role --role-name AWS ServiceRoleForElastiCache
   ```

   CLI オペレーションでロールを参照するにはARN、 ではなくロール名を使用します。例えば、ロールにARN次の がある場合arn:aws:iam::123456789012:role/myrole、ロールを として参照しますmyrole。
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 AWS CLI に を使用しますupdate-role-description。

   Linux、macOS、Unix の場合:

   ```
   $ aws iam update-role-description \
       --role-name AWS ServiceRoleForElastiCache \
       --description "new description"
   ```

   Windows の場合:

   ```
   $ aws iam update-role-description ^
       --role-name AWS ServiceRoleForElastiCache ^
       --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)

を使用してIAMAPI、サービスにリンクされたロールの説明を編集できます。

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) ロールの現在の説明を表示するには、 IAMAPIオペレーションを使用します。GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWS ServiceRoleForElastiCache
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. ロールの説明を更新するには、 IAMAPIオペレーションを使用します。UpdateRoleDescription.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWS ServiceRoleForElastiCache
      &Version=2010-05-08
      &Description="New description"
   ```

Amazon のサービスにリンクされたロールの削除 ElastiCache

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、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、削除する前に、サービスにリンクされた役割をクリーンアップする必要があります。

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前に、まずロールにリソース (クラスターまたはレプリケーショングループ) が関連付けられていないことを確認します。

サービスにリンクされたロールにIAMコンソールでアクティブなセッションがあるかどうかを確認するには

1. にサインイン AWS Management Console し、 でIAMコンソールを開きますhttps://console.aws.amazon.com/iam/。
2. IAM コンソールのナビゲーションペインで、ロール を選択します。次に、 AWS ServiceRoleForElastiCache ロールの名前 (チェックボックスではない) を選択します。
3. 選択したロールの 概要 ページで、アクセスアドバイザー タブを選択します。
4. アクセスアドバイザー タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

を必要とする Amazon ElastiCache リソースを削除するには AWS ServiceRoleForElastiCache

- クラスターを削除するには、以下を参照してください。
  - の使用 AWS Management Console
  - AWS CLI を使用して ElastiCache クラスターを削除する
  - の使用 ElastiCache API
- 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. にサインイン AWS Management Console し、 でIAMコンソールを開きますhttps://console.aws.amazon.com/iam/。
2. IAM コンソールのナビゲーションペインで、ロール を選択します。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、最後にアクセスしたサービスデータを確認します。このデータには、選択した各ロールが最後に AWS サービスにアクセスした日時が表示されます。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 コンソールの通知を見て、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、削除のためにロールを送信すると、削除タスクが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 View details] (詳細を表示) または View Resources] (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)

から IAM オペレーションを使用して AWS Command Line Interface 、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。このコマンドは、アカウントのロールとその Amazon リソースネーム (ARNs) を一覧表示します。

```
$ aws iam get-role --role-name role-name
```

CLI オペレーションでロールを参照するにはARN、 ではなくロール名を使用します。例えば、ロールに ARN がある場合`arn:aws:iam::123456789012:role/myrole`、ロールを と呼びます**`myrole`**。

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)

を使用してIAMAPI、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、 を呼び出します。DeleteServiceLinkedRole。 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. 削除のステータスを確認するには、 を呼び出します。GetServiceLinkedRoleDeletionStatus。 リクエストで、 を指定しますDeletionTaskId。

   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

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) にアタッチするアクセスコントロールポリシーと書き込みアクセス許可IAMポリシーを設定するときは、次の表を参照として使用します。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 がない場合ARN、ポリシー内のリソースはワイルドカード (*) です。

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特定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、 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。 IAM

> **Note**
>
> アクションを指定するには、`elasticache:`プレフィックスの後にAPIオペレーション名 (例: `elasticache:DescribeCacheClusters`) 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 によって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、、PCIFed 、 などの複数のコンプライアンスプログラムの一環としてRAMP、 AWS サービスのセキュリティと AWS コンプライアンスを評価しますHIPAA。

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、AWS のサービス「コンプライアンスプログラムによるスコープ」の」の「」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することができます AWS Artifact。詳細については、「」の AWS Artifact」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、データの機密性、会社のコンプライアンス目的、および適用される法律と規制によって決まります。 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。
- Amazon Web Services HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ – このホワイトペーパーでは、企業が AWS を使用して HIPAAの対象となる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**
>
> すべての AWS のサービス がHIPAA対象となるわけではありません。詳細については、HIPAA「対象サービスリファレンス」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、お客様の業界とロケーション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) など) 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ています。
- AWS Config 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、リソース設定が内部プラクティス、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 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モニタリングすることで、、、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 を検出できます。 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出要件を満たすことでDSS、 PCI などのさまざまなコンプライアンス要件に対応するのに役立ちます。

- 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、リスクと規制や業界標準へのコンプライアンスの管理を簡素化できます。

詳細情報

AWS クラウドコンプライアンスに関する一般的な情報については、以下を参照してください。

- FIPS サービス別のエンドポイント
- のサービス更新 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 のレジリエンス ElastiCache

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および冗長性の高いネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

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 は、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするいくつか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 実装を計画するときは、障害がアプリケーションと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

- Memcached 実行時の障害を軽減する
- Valkey または Redis の実行時の障害の軽減 OSS
- レコメンデーション

Memcached 実行時の障害を軽減する

Memcached エンジンを実行する場合に、障害の影響を最小にするためのオプションとして次のものがあります。障害の軽減に対処する方法には、ノードの障害の軽減とアベイラビリティーゾーンの障害の軽減の 2 つのタイプがあります。

ノードの障害の軽減

サーバーレスキャッシュは、レプリケートされたマルチ 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。独自設計型クラスターにおけるノードの障害の影響を軽減するには、キャッシュデータをより多くのノードに広げます。独自設計型クラスターがレプリケーションをサポートしていないため、ノードの障害によって必ずクラスターからある程度のデータが失われます。

Memcached クラスターを作成するときは、特別なリクエストにより、1～60 ノード以上で作成できます。大量のノード間でデータのパーティションを行うと、ノードで障害が発生した場合のデータの損失が小さくなります。たとえば、10 のノード間でデータのパーティションを行うと、単一のノードに約 10% のキャッシュデータが保存されることになります。この場合、ノードの障害が起きるとキャッシュの約 10% が失われ、代替ノードが作成されプロビジョニングされたときに置き換える必要があります。同じデータがより大きな 3 つのノードにキャッシュされている場合は、ノードの障害によってキャッシュされたデータの約 33% が失われます。

Memcached クラスターのノード数を指定する方法の詳細については、「Memcached クラスター (CLI) の作成 (コンソール)」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、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配置します。万一 AZ に障害が発生した場合、他のにキャッシュされたデータではなく、その AZ にキャッシュされたデータが失われますAZs。

なぜ大量のノードが必要ですか。

自分のリージョンに 3 つのアベイラビリティーゾーンのみがある場合、AZ で障害が発生すればデータの約 3 分の 1 を失うことになるので、なぜ 3 つ以上のノードが必要なのですか。

これはいい質問です。当社では、ノードの障害とアベイラビリティーゾーンの障害の 2 つの明確な障害を軽減しようとしてきました。ご指摘のとおり、データが各アベイラビリティーゾーンにまたがっており、ゾーンの 1 つで障害が発生した場合は、ノード数に関係なくその AZ でキャッシュされたデータのみが失われます。ただしノードで障害が発生した場合は、できるだけ多くのノードがあったほうが、失われるデータの割合が減ります。

クラスターのノード数を決定する「魔法の公式」はありません。データ損失の影響と障害が発生する可能性とコストを考慮して、個別に判断を下す必要があります。

Memcached クラスターのノード数を指定する方法の詳細については、「Memcached クラスター (CLI) 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」を参照してください。

Valkey または Redis の実行時の障害の軽減 OSS

Valkey または Redis OSS エンジンを実行する場合、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抑えるために、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。

Valkey または Redis OSSノード障害がセルフ設計クラスターに与える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

失敗の軽減: Valkey または Redis OSS レプリケーショングループ

Valkey または Redis OSSレプリケーショングループは、1 つのプライマリノードで構成されており、アプリケーションは 1 ~ 5 の読み取り専用レプリカノードとの間で読み取りと書き込みの両方を行うことができます。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。

Valkey または Redis OSS マルチ AZ

マルチ AZ は、Valkey または Redis OSSレプリケーショングループで有効にできます。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、レプリケーション遅延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、失敗したプライマリの AZ でリードレプリカをスピンアップします。
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、「Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS」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めません。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、レプリケーショングループのノードを異なるアベイラビリティーゾーン () で起動することをお勧めしますAZs。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、複数のにノードを見つけた場合AZs、AZ が失敗すると、その AZ 内のノードのみが失われます。

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継続します。

Valkey ノードまたは Redis OSSノードのアベイラビリティーゾーンの指定については、「」を参照してくださいValkey (クラスターモードが無効) クラスターの作成 (コンソール) 。

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、「のリージョンとアベイラビリティーゾーンの選択 ElastiCache」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

Valkey または Redis を使用する際のノード障害の影響を最小限に抑えるためにOSS、実装では各シャードに複数のノードを使用し、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。

Valkey または Redis の自己設計型クラスターではOSS、プライマリノードが失敗した場合に ElastiCache がレプリカに自動的にフェイルオーバーするように、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。

Memcached を実行してノード間でデータを仕切っている場合は、ノードの数を増やすほど 1 つのノードで障害が発生した場合のデータの損失をより小さくすることができます。

アベイラビリティーゾーンの障害の影響を最小限に抑える

アベイラビリティー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ーゾーンでノードを起動することをお勧めします。ノードを均等に分散AZsすることで、万一 AZ 障害が発生した場合の影響を最小限に抑えることができます。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

Valkey または Redis を実行している場合はOSS、上記に加えて、クラスターの定期的なバックアップをスケジュールす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。

のインフラストラクチャセキュリティ AWS ElastiCache

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に記載されている AWS グローバルネットワークセキュリティ手順によって保護されています。

AWS 公開されたAPI呼び出しを使用して、ネットワーク ElastiCache 経由で 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。1.3 TLS 以降をお勧めします。また、クライアントは、エフェメラルディフィヘルマン (PFS) や楕円曲線エフェメラルディフィヘルマン () など、完全なフォワードシークレット (DHE) を持つ暗号スイートもサポートする必要がありますECDHE。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

さらに、リクエストは、アクセスキー ID とプリンシIAM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。

のサービス更新 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、サービスの更新が利用可能になったときに適用します。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。独自設計のクラスターでは、 ElastiCache がこれらの更新を適用できるように、事前定義されたメンテナンスウィンドウを設定します。ただし、場合によっては、このアプローチが厳格すぎて、ビジネスフローが制限される可能性もあります。

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す。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。

トピック

- 独自設計クラスターのサービス更新の管理

独自設計クラスターのサービス更新の管理

ElastiCache 独自設計クラスターのサービス更新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 1 つ以上のセルフデザインクラスターがある場合は、更新がリリースされると、E メール、、パーソナルヘルスダッシュボード (PHD) SNS、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、 ElastiCache コンソールの Service Updates ページにも表示

されます。このダッシュボードを使用すると、フリー ElastiCache トのすべてのサービス更新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 クラスターに常に up-to-date最新のセキュリティパッチが適用されていることを確認するために、タイプ Security-Update の更新を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。

サービスの更新の適用

フリートに対するサービスの更新の適用は、更新が 使用可能 ステータスになってから開始することができます。サービスの更新は累積的です。つまり、未適用の更新も最新の更新に含まれます。

サービスの更新で自動更新が有効になっている場合、利用可能になったときにアクションを実行しないことを選択できます。 ElastiCache は、自動更新の開始日 の後に、クラスターの今後のメンテナンスウィンドウの 1 つに更新を適用するようにスケジュールします。更新のステージごとに、関連する通知を受け取ります。

> **Note**
>
> ステータスが 使用可能または スケジュール済み であるサービスの更新だけを適用できます。

該当する ElastiCache クラスターに対するサービス固有の更新の確認と適用の詳細については、「」を参照してくださいコンソールを使用したサービスの更新の適用。

1 つ以上の ElastiCache クラスターで新しいサービス更新が利用可能な場合は、 ElastiCache コンソール、API、または を使用して更新 AWS CLI 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. にサインイン AWS Management Console し、 で Amazon ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。

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: クラスター上のサービス更新のステータス。以下のいずれかです。
     - 使用可能: 必要なクラスターでこの更新が利用可能です。

- 進行中: このクラスターに更新を適用しています。
- スケジュール済み: 更新日がスケジュールされています。
- 完了: 更新が正常に適用されました。完了ステータスのクラスターは、完了後 7 日間表示されます。

ステータスが 使用可能または スケジュール済み (スケジュール済み) であるクラスターのいずれかまたはすべてを選択してから、今すぐ適用を選択した場合、更新がそれらのクラスターに適用され始めます。

AWS CLIを使用してサービスの更新を適用する

サービスの更新が利用可能であるという通知を受け取ったら、 AWS CLIを使用してそれらの更新を確認し、適用することができます。

- 利用可能なサービスの更新の説明を取得するには、次のコマンドを実行します。

  ```
  aws 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
  ```

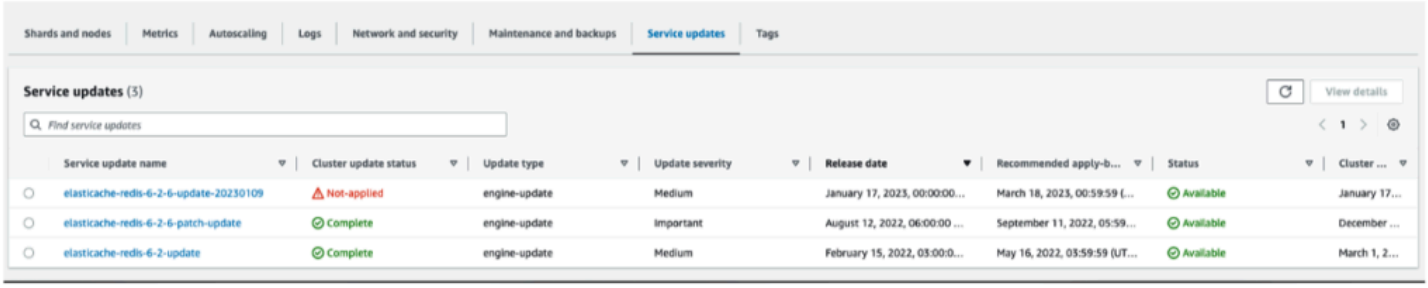
  詳細については、「」を参照してくださいbatch-apply-update-action。

AWS コンソールを使用して最新の Service Update が適用されていることを確認する

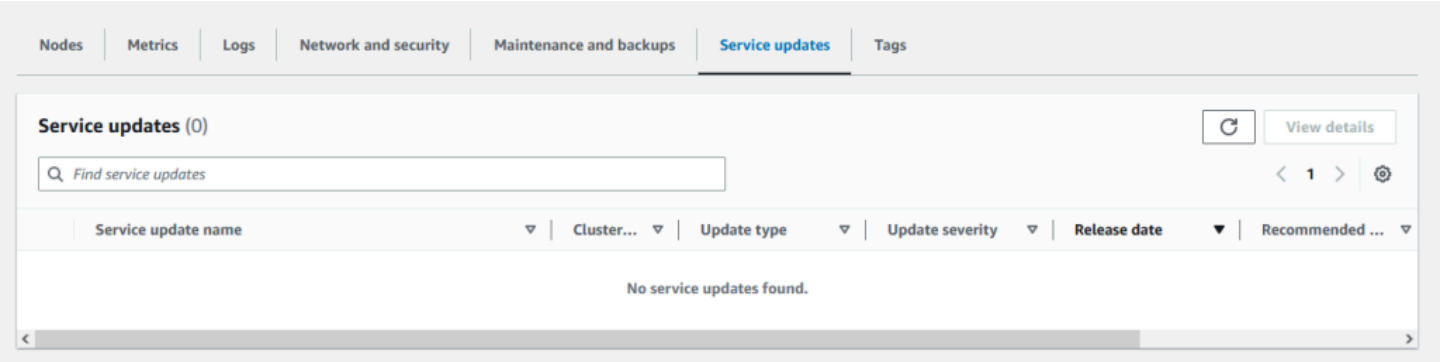
(ElastiCache Redis OSS) クラスターが最新のサービス更新を実行していることを確認するには、次の手順に従います。

1. Redis クラスターページで該当するOSSクラスターを選択する
2. ナビゲーションペインでサービスの更新を選択すると、そのクラスターに該当するサービスの更新が表示されます。

コンソールにサービス更新のリストが表示された場合は、サービス更新を選択し、今すぐ適用 を選択できます。



コンソールに「サービス更新が見つかりません」と表示される場合は、 ElastiCache (Redis OSS) クラスターに最新のサービス更新が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 コンソールは ElastiCache ダッシュボードの上部に更新の表示/停止タブを表示します。
- 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。

の使用 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。

一般的な脆弱性と露出 (CVE): で対処されるセキュリティの脆弱性 ElastiCache

Common Vulnerabilities and Exposures (CVE) は、一般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リストです。各エントリは、識別番号、説明、および少なくとも 1 つのパブ

リックリファレンスを含むリンクです。このページでは、 で対処されたセキュリティ脆弱性のリストを確認できます ElastiCache。

既知の脆弱性から保護するために、常に最新の ElastiCache Valkey、Redis、OSSまたは ElastiCache Memcached 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 Serverless Cache を操作すると、CVE修正は自動的にキャッシュに適用されます。Valkey または Redis で自己設計型クラスターを操作するとOSS、 はPATCHコンポーネントを ElastiCache 公開します。例えば、 ElastiCache (Redis OSS) バージョン 6.2.6 を使用する場合、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。PATCH 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用です。

次の表を使用して、特定のバージョンの ElastiCache Valkey と Redis に特定のセキュリティ脆弱性の修正OSSがあるかどうかを確認できます。 ElastiCache Valkey または Redis OSSクラスターでセキュリティ修正なしでバージョンを実行している場合は、以下の表を参照してアクションを実行します。修正を含むより新しい ElastiCache Valkey または Redis OSSバージョンにアップグレードするか、修正を含む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更新が適用されていることを確認してください独自設計クラスターのサービス更新の管理。サポートされている ElastiCache エンジンのバージョンとアップグレード方法の詳細については、「」を参照してくださいでのエンジンバージョンとアップグレード ElastiCache。

> **Note**
>
> - CVE が ElastiCache バージョンでアドレス指定されている場合、新しいバージョンでもアドレス指定されることを意味します。例えば、 ElastiCache (Redis OSS) バージョン 6.0.5 で脆弱性に対処した場合、バージョン 6.2.6、7.0.7、および 7.1 では引き続き脆弱性が解決されます。
> - 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、 ElastiCache (Redis OSS) バージョンを実行している ElastiCache (Redis OSS) クラスターに最新のサービス更新を適用する必要があることを示しています。クラスターが実行中の ElastiCache (Redis OSS) バージョンに最新のサービス更新が適用されていることを確認する方法の詳細については、「」を参照してください独自設計クラスターのサービス更新の管理。

ElastiCache (Redis OSS) バージョン	CVEs アドレス指定済み
Redis OSS 6.0.5	CVE-2022-24735*、CVE-2022-24736*

ElastiCache (Redis OSS) バージョン	CVEs アドレス指定済み
Redis OSS 6.2.6	CVE-2022-24834*、CVE-2022-35977*、CVE-2022-36021*、CVE-2022-24735、CVE-2022-24736
Redis OSS 7.0.7	CVE-2023-41056*、CVE-2022-24834*、CVE-2022-35977、CVE-2022-36021、CVE-2022-24735、CVE-2022-24736
Redis OSS 7.1.0	CVE-2023-41056、CVE-2022-24834、CVE-2022-35977、CVE-2022-36021、CVE-2022-24735、CVE-2022-24736

Amazon でのログ記録とモニタリング ElastiCache

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、キャッシュパフォーマンスをモニタリングするために Amazon CloudWatch Logs に発行されるメトリクスを生成します。さらに、キャッシュリソースに大きな変更が発生した場合 (新しいキャッシュが作成された場合やキャッシュが削除された場合など) にイベント ElastiCache を生成します。

トピック

- Valkey と Redis のサーバーレスメトリクスとイベント OSS
- Memcached のサーバーレスメトリクスとイベント
- を使用した Amazon ElastiCache API コールのログ記録 AWS CloudTrail
- ElastiCache イベントの Amazon SNSモニタリング
- ログ配信
- CloudWatch メトリクスの使用のモニタリング

- を使用した Amazon ElastiCache API コールのログ記録 AWS CloudTrail

Valkey と Redis のサーバーレスメトリクスとイベント OSS

ElastiCache は、サーバーレスキャッシュを操作するときにモニタリングするためのさまざまなメトリクスとイベントを提供します。これには、Amazon 経由で取り込むことができる CloudWatch メトリクス、コマンドレベルのメトリクス、イベントログが含まれます EventBridge。

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
- Valkey と Redis の独自設計クラスターメトリクスとイベント OSS

サーバーレスキャッシュメトリクス

`AWS/ElastiCache` 名前空間には、Valkey または Redis OSSサーバーレスキャッシュの以下の CloudWatch メトリクスが含まれます。

Valkey または Redis のメトリクスコード OSS

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	キャッシュで実行されたリクエストによって消費された ElastiCacheProcessingUnits (ECPUs) の合計数	カウント
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: `cache_hits /(cache_hits + 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	ワークロードがスケーリング ElastiCache 可能よりも速くスケーリングされた ElastiCache	カウント

メトリクス	説明	単位
	ために によってスロットリングされたリクエストの数。	
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
CurrVolatileItems	を含むキャッシュ内の項目の数TTL。	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	期限切れのIAM認証済み Valkey または Redis OSS接続の合計数。IAM による認証 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認証済み Valkey または Redis OSSAUTHまたはHELLOリクエストの総数。IAM による認証 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

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。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
AuthenticationFailures	AUTH コマンドOSSを使用して Valkey または 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、次のコマンドレベルのメトリクスも出力します。各コマンドタイプについて、 はコマンドの合計数と、そのコマンドタイプによってECPUs消費された数を ElastiCache 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	ECPUs 評価ベースのコマンドによって消費されます。	カウント

メトリクス	説明	単位
GeoSpatialBasedCmds	地理空間ベースのコマンドの総数。これは Valkey または Redis OSS コマンド統計から派生します。これは、geoadd、geodist、geohash、geopos、georadius、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。	カウント
GeoSpatialBasedCmdsECPUs	ECPUs 地理空間ベースのコマンドによって消費されます。	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OSSコマンド (get、hget、scard、lrange など) を合計して、Valkey または Redis コマンド統計から派生します。	カウント
GetTypeCmdsECPUs	ECPUs 読み取りコマンドによって消費されます。	カウント
HashBasedCmds	ハッシュベースのコマンドの総数。これは、1 つ以上のハッシュ (hget、hkeys、hvals、hdel など) に作用するすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
HashBasedCmdsECPUs	ECPUs ハッシュベースのコマンドによって消費されます。	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OSSコマンド (pfadd、pfcount、pfmerge など) を合計して、Valkey または Redis コマンド統計から派生します。	カウント
HyperLogLogBasedCmdsECPUs	ECPUs HyperLogLogベースのコマンドによって消費されます。	カウント
JsonBasedCmds	読み取りJSONコマンドと書き込みコマンドの両方を含むコマンドの合計数。これは、JSONキーで動作するすべてのOSSコマンドを合計することで、Valkey または Redis JSON コマンド統計から派生します。	カウント
JsonBasedCmdsECPUs	ECPUs 読み取りJSONコマンドと書き込みコマンドの両方を含むすべてのコマンドによって消費されます。	カウント
JsonBasedGetCmds	JSON 読み取り専用コマンドの合計数。これは、JSONキーで動作するすべてのJSON読み取りOSSコマンドを合計することで、Valkey または Redis コマンド統計から派生します。	カウント

メトリクス	説明	単位
JsonBasedGetCmdsECPUs	ECPUs JSON読み取り専用コマンドによって消費されます。	カウント
JsonBasedSetCmds	JSON 書き込みコマンドの合計数。これは、JSONキーで動作するすべてのJSON書き込みOSSコマンドを合計することで、Valkey または Redis コマンド統計から派生します。	カウント
JsonBasedSetCmdsECPUs	ECPUs JSON書き込みコマンドによって消費されます。	カウント
KeyBasedCmds	キーベースのコマンドの総数。これは、複数のデータ構造 (del、期限切れ、名前変更など) にわたって 1 つ以上のキーで動作するすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
KeyBasedCmdsECPUs	ECPUs キーベースのコマンドによって消費されます。	カウント
ListBasedCmds	リストベースのコマンドの総数。これは、1 つ以上のリスト (lindex、lrange、lpush、ltrim など) で動作するすべてのコマンドを合計することで、Valkey または Redis コマンドOSS統計から派生します。	カウント

メトリクス	説明	単位
ListBasedCmdsECPUs	ECPUs はリストベースのコマンドによって消費されます。	カウント
NonKeyTypeCmds	キーベースではないコマンドの合計数。これは、acl、dbsize、または info など、キーに作用しないすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
NonKeyTypeCmdsECPUs	ECPUs non-key-based コマンドによって消費されます。	カウント
PubSubBasedCmds	pub/sub 機能のコマンドの総数。これは、psubscribe、publish、pubsub、punsubscribe、ssubscribe、sunsubscribe、spublish、subscribe、および unsubscribe の機能に使用されるすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
PubSubBasedCmdsECPUs	ECPUs pub/sub-based コマンドによって消費されます。	カウント

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 つ以上のセット (カード、sdiff、sadd、sunion など) で動作するすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
SetBasedCmdsECPUs	ECPUs セットベースのコマンドによって消費されます。	カウント
SetTypeCmds	書き込みタイプのコマンドの合計数。これは、データ (set、hset、sadd、lpop など) で動作するすべてのミュータティブタイプのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
SetTypeCmdsECPUs	ECPUs 書き込みコマンドによって消費されます。	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 つ以上のソートされたセット (zcount、zrange、zrank、zadd など) で動作するすべてのコマンドを合計することで、Valkey または Redis コマンドOSS統計から派生します。	カウント
SortedSetBasedCmdsECPUs	ECPUs ソートベースのコマンドによって消費されます。	カウント

メトリクス	説明	単位
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 (strlen、setex、setrange など) で動作するすべてのコマンドを合計することで、Valkey または Redis OSS コマンド統計から派生します。	カウント
StringBasedCmdsECPUs	ECPUs 文字列ベースのコマンドによって消費されます。	カウント
StreamBasedCmds	ストリームベースのコマンドの総数。これは、1 つ以上のストリームデータ型 (xrange、xlen、xadd、xdel など) で動作するすべてのコマンドを合計することで、Valkey または Redis コマンドOSS統計から派生します。	カウント
StreamBasedCmdsECPUs	ECPUs ストリームベースのコマンドによって消費されます。	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ログに記録します。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 アクションを使用して、ログからイベントを簡単に取得できます`DescribeEvents`。

Amazon を使用して、 ElastiCache イベントのモニタリング、取り込み、変換、およびアクションを選択できます EventBridge。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. にサインイン AWS Management Console し、 で ElastiCache コンソールを開きます。 https://console.aws.amazon.com/elasticache/
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 [イベント] 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時刻、および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)

を使用して ElastiCache イベントのリストを生成するには AWS CLI、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>

詳細タイプ	説明	単位	ソース	メッセージ
				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。
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のに十分な空き IP アドレスがありません。
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。

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 エンドポイントの作成にクォータ制限に達しました。
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。
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 Valkey または Redis OSSバージョンを持つユーザーは、ユーザーグループ <user-group-name> に存在します。

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 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	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。 失敗の理由: • 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。 • 指定された RDBファイルにはサポートされていないバージョンがあります。

サーバーレスキャッシュ更新イベント (バルキーまたは Redis OSS)

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	SecurityGroups がキャッシュ

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				<cache-name> 用に更新されました。
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 Valkey または Redis OSS バージョンを持つユーザーは、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> の更新が失敗 SecurityGroups 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	アクセス許可が不十分なため、キャッシュ <cache-name> failed. SecurityGroups update の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。SecurityGroups SecurityGroups が無効なため、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 (バルキーまたは Redis OSS)

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の使用制限イベント (バルキーまたは Redis OSS)

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

詳細タイプ	説明	単位	ソース	メッセージ
				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。
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、キャッシュの上限を更新できませんでした。
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 (Valkey または Redis OSS)

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	serverless-cache-snapshot	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	serverless-cache-snapshot	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> を作成できませんでした。 失敗理由メッセージ: • 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

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				• 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	serverless-cache-snapshot	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。 既定の理由: • 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	serverless-cache-snapshot	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。ElastiCache にはバケットに対する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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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	serverless-cache-snapshot	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。

Valkey と Redis の独自設計クラスターメトリクスとイベント OSS

ElastiCache は、Valkey および Redis を操作するときに、独自設計のクラスターをモニタリングするためのさまざまなメトリクスとイベントを提供しますOSS。これには、 および AWS CLI Amazon Simple Notification Service () を介して利用可能なホストレベルのメトリクス、コマンドレベルのメトリクス、イベントログが含まれますSNS。

トピック

独自設計型クラスターのメトリクス

クラスターをセルフ設計すると、 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各ノードレベルでメトリクスを ElastiCache 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、「Valkey と Redis のメトリクス OSS」を参照してください。

独自設計クラスターのイベント (バルキーおよび Redis OSS)

ElastiCache は、独自設計のキャッシュに関連するイベントをログに記録します。独自設計のクラスターを使用する場合は、 ElastiCache コンソール、、 AWS CLIまたは Amazon Simple Notification Service () を使用してクラスターイベントを表示できますSNS。独自設計のクラスターイベントは Amazon に公開されません EventBridge。

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 を使用して、ログからイベントを簡単に取得できます DescribeEvents。

ElastiCache イベントの表示 (コンソール)

次の手順では、 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イベントを表示するには

1. にサインイン AWS Management Console し、 で ElastiCache コンソールを開きます。 https://console.aws.amazon.com/elasticache/
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 [イベント] 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時刻、およびイベントの説明が表示されます。フィルターを使用して、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)

を使用して ElastiCache イベントのリストを生成するには AWS CLI、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。

次の ElastiCache イベントは Amazon SNS通知をトリガーします。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 : cache-cluster`	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: 空き IP アドレスが不足AddCacheNodeFailedしているため	ElastiCache:AddCacheNodeFailed : *cluster-name*	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 : *cluster-name*	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 *cluster-name-0001-005*	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: 互換性のないネットワーク状態CacheClusterProvisioningFailed のため	ElastiCache:CacheClusterProvisioningFailed : *cluster-name*	存在しない仮想プライベートクラウド () に新しいキャッシュクラスターを起動しようとしましたVPC。
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : *cluster-name*	キャッシュクラスターのスケーリングが正常に完了しました。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	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 : cluster-name`	以下のいずれかのイベントが発生しました。 • 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。 • キャッシュクラスターに関連付けられたキャッシュEC2セキュリティグループのいずれかで、1 つ以上の新しいセキュリティグループが承認されています。 • 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ているキャッシュセキュリティグループのいずれか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 : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストの劣化または到達不能を検出し、キャッシュノードの置き換えを開始しました。 **Note** 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。 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、 がキャッシュノードを ElastiCache 置き換えた後でも、キャッシュノードの使用を停止する場合があります。この場合、アプリケーションはこのイベントが発生したときにサーバーリストを更新する必要があります。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストの劣化または到達不能を検出し、キャッシュノードの交換を完了しました。 **Note** 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。 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、 がキャッシュノードを ElastiCache 置き換えた後でも、キャッシュノードの使用を停止する場合があります。この場合、アプリケーションはこのイベントが発生したときにサーバーリストを更新する必要があります。

イベント名	メッセージ	説明
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	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（バルキーまたは Redis OSSのみ）	`ElastiCache:CertificateRenewalComplete`	Amazon CA 証明書が正常に更新されました。
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete : cluster-name`	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 : cluster-name`	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（バルキーまたは Redis OSSのみ）	`ElastiCache:FailoverComplete : mycluster`	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 : cluster-name-0001-005`	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 : cluster-name-0003-004`	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 : cluster-name`	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 : cluster-name`	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、「ノードの交換 (Valkey と Redis OSS)」を参照してください。
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	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、「ノードの交換 (Valkey と Redis OSS)」を参照してください。
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	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。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	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。
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（バルキーまたは Redis OSSのみ）	`ElastiCache:SnapshotComplete : cluster-name`	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（バルキーまたは Redis OSSのみ）	`SnapshotFailed : cluster-name`	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。

Memcached のサーバーレス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

このセクションでは、Memcached 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について説明します。

`AWS/ElastiCache` 名前空間には、Memcached サーバーレスキャッシュの以下の 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	キャッシュで実行されたリクエストによって消費された ElastiCacheProcessingUnits (ECPUs) の合計数	カウント
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

メトリクス	説明	単位
CurrConnections	キャッシュへのクライアント接続の数。	カウント
ThrottledCmds	ワークロードがスケーリング ElastiCache 可能よりも速くスケーリングされた ElastiCache ために 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
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
Reclaimed	キャッシュによって期限切れになったキーの数。	カウント

コマンドレベルメトリクス

ElastiCache は、次の Memcached コマンドレベルのメトリクスも出力します。

メトリクス	説明	単位
CmdGet	キャッシュが受信した get コマンドの数。	カウント

メトリクス	説明	単位
CmdSet	キャッシュが受信した set コマンドの数。	カウント
CmdTouch	キャッシュが受信した touch コマンドの数。	カウント
GetHits	キャッシュが受信し、リクエストされたキーが見つかった get リクエストの数。	カウント
GetMisses	キャッシュが受信したが、リクエストされたキーが見つからなかった get リクエストの数。	カウント
IncrHits	キャッシュが受信し、リクエストされたキーが見つかったインクリメントリクエストの数。	カウント
IncrMisses	キャッシュが受信したが、リクエストされたキーが見つからなかったインクリメントリクエストの数。	カウント
DecrHits	キャッシュが受信し、リクエストされたキーが見つかったデクリメントリクエストの数。	カウント
DecrMisses	キャッシュが受信したが、リクエストされたキーが見つからなかったデクリメントリクエストの数。	カウント

メトリクス	説明	単位
DeleteHits	キャッシュが受信し、リクエストされたキーが見つかった削除リクエストの数。	カウント
DeleteMisses	キャッシュが受信したが、リクエストされたキーが見つからなかった削除リクエストの数。	カウント
TouchHits	タッチされて新しい有効期限を与えられたキーの数。	カウント
TouchMisses	タッチされたが見つからなかったキーの数。	カウント
CasHits	キャッシュが受信し、リクエストされたキーが見つかって cas 値が一致した cas リクエストの数。	カウント
CasMisses	キャッシュが受信したが、リクエストされたキーが見つからない cas リクエストの数。	カウント
CasBadval	キャッシュが受信したが、その cas 値と格納されている cas 値が一致しない cas リクエストの数。	カウント
CmdFlush	キャッシュが受信した flush コマンドの数。	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ログに記録します。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 アクションを使用して、ログからイベントを簡単に取得できます`DescribeEvents`。

Amazon を使用して、 ElastiCache イベントのモニタリング、取り込み、変換、およびアクションを選択できます EventBridge。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. にサインイン AWS Management Console し、 で ElastiCache コンソールを開きます。 https://console.aws.amazon.com/elasticache/
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 [イベント] 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時刻、および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)

を使用して ElastiCache イベントのリストを生成するには AWS CLI、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のに十分な空き IP アドレスがありません。
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 エンドポイントを作成するため

詳細タイプ	説明	単位	ソース	メッセージ
				のクォータ制限に達しました。
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。

サーバーレスキャッシュ更新イベント (Memcached)

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	SecurityGroups がキャッシュ <cache-name> 用に更新されました。
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> の更新が失敗 SecurityGroups 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	アクセス許可が不十分な

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				め、キャッシュ <cache-name> の更新が失敗 SecurityGroups 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	が無効なため、キャッシュ <cache-name> failed. SecurityGroups update の更新 SecurityGroups に失敗しました。

サーバーレスキャッシュ削除イベント (Memcached)

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の使用制限イベント (Memcached)

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。

詳細タイプ	説明	単位	ソース	メッセージ
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。
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、キャッシュの上限を更新できませんでした。

サーバーレスキャッシュスナップショットイベント (Memcached)

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	serverless-cache-snapshot	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	serverless-cache-snapshot	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

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				キー <key-id> でスナップショット <snapshot-name> を作成できませんでした。 失敗理由メッセージ: • 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

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失敗	キャッシュ ARN スナップショット ARN	失敗	serverless-cache-snapshot	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。 既定の理由: • 内部エラーのため
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	serverless-cache-snapshot	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。ElastiCache にはバケットに対する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。

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	serverless-cache-snapshot	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。
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	serverless-cache-snapshot	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。

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	serverless-cache-snapshot	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。
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	serverless-cache-snapshot	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。

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	serverless-cache-snapshot	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。
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	serverless-cache-snapshot	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。

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	serverless-cache-snapshot	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。
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	serverless-cache-snapshot	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。

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	serverless-cache-snapshot	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。

を使用した Amazon ElastiCache API コールのログ記録 AWS CloudTrail

Amazon ElastiCache は AWS CloudTrail、Amazon のユーザー、ロール、または AWS サービスによって実行されたアクションの記録を提供するサービス と統合されています ElastiCache。は、Amazon ElastiCache コンソールからのAPI呼び出しや Amazon オペレーションへのコード呼び出しを含む、Amazon のすべての呼び出しをイベント ElastiCache として CloudTrail キャプチャします ElastiCache API。証跡を作成する場合は、Amazon の CloudTrail イベントを含む、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます ElastiCache。 Amazon S3 証跡を設定しない場合でも、 CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、Amazon に対して行われたリクエスト ElastiCache、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、その他の詳細を確認できます。

の詳細については CloudTrail、AWS CloudTrail 「 ユーザーガイド」を参照してください。

の Amazon ElastiCache 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 はアカウントで有効になります。Amazon でアクティビティが発生すると ElastiCache、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴 での CloudTrail イベントの表示」を参照してください。

Amazon のイベントなど、 AWS アカウント内のイベントの継続的な記録については ElastiCache、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 パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータをさらに分析して処理するように、他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail サポートされているサービスと統合
- の Amazon SNS Notifications の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

すべての Amazon ElastiCache アクションは によってログに記録 CloudTrail され、 ElastiCache API リファレンス に記載されています。例えば、 を呼び出すDescribeCacheClusterとCreateCacheCluster、 ModifyCacheClusterアクションは CloudTrail ログファイルにエントリを生成します。

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。アイデンティティ情報は、以下を判別するのに役立ちます:

- リクエストがルート認証情報またはIAMユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity「 要素」を参照してください。

Amazon ElastiCache ログファイルエントリについて

証跡は、指定した Amazon S3 バケットへのログファイルとしてイベントを配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは、任意の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、アクションの日付と時刻、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリックAPIコールの順序付けられたスタックトレースではないため、特定の順序で表示されません。

次の例は、 `CreateCacheCluster`アクションを示す CloudTrail ログエントリを示しています。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:00:35Z",
    "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 ログエントリを示しています。すべての Amazon ElastiCache Describe 呼び出し (Describe*) では、 ResponseElementsセクションが削除され、 として表示されますnull。

```
{
    "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"
}
```


次の例は、ModifyCacheClusterアクションを記録する CloudTrail ログエントリを示しています。


```
{
    "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 イベントの Amazon SNSモニタリング

クラスターで重大なイベントが発生すると、 は特定の Amazon SNSトピックに通知 ElastiCache を送信します。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) を使用して、重要なクラスターイベントの通知を送信する ElastiCache ように を設定できますSNS。これらの例では、通知を受信するように Amazon SNSトピックの Amazon リソースネーム (ARN) でクラスターを設定します。

> **Note**
>
> - このトピックでは、Amazon にサインアップSNSし、Amazon SNSトピックをセットアップして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。
> - デフォルトでは、 は、現在の指定されたグループだけでなく、リージョン内のすべてのグループAPI `modify-replication-group`に影響します。リージョン内の 1 つの特定のグループを他のグループとは異なる方法で設定する場合は、 `--notification-topic-arn`オプションを使用して、そのグループの別のトピックを作成できます。

Amazon SNSトピックの追加

以下のセクションでは、 AWS コンソール、、 AWS CLIまたは を使用して Amazon SNSトピックを追加する方法を示します ElastiCache API。

Amazon SNSトピックの追加 (コンソール)

次の手順では、クラスターに Amazon SNSトピックを追加する方法を示します。Valkey または Redis を使用してステップ 2 でレプリケーショングループの Amazon SNSトピックOSSを追加する場合、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ます。次に、同じ残りのステップに従います。

> **Note**
>
> このプロセスは、Amazon SNSトピックを変更するためにも使用できます。

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. 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. クラスター で、Amazon SNSトピック を追加または変更するクラスターを選択しますARN。
3. Modify を選択します。
4. SNS 「通知用トピック」の「クラスターの変更」で、追加するSNSトピックを選択するか、「手動ARN入力」を選択し、Amazon SNSトピックARNの「」を入力します。
5. Modify を選択します。

Amazon SNSトピックの追加 (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 を使用しますmodify-cache-cluster。

次のコード例では、my-cluster に Amazon SNSトピック arn を追加します。

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、「」を参照してくださいmodify-cache-cluster。

Amazon SNSトピックの追加 (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、次のパラメータを使用して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.amazon.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、「」を参照してくださいModifyCacheCluster。

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)

を使用して Amazon SNS通知を無効にするには 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. Memcached 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、[Memcached] を選択します。

   Valkey または Redis を実行しているクラスターのリストを表示するにはOSS、ナビゲーションペインで Valkey または Redis OSS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 を選択します。
5. SNS 「通知トピック」の「クラスターの変更」で、「通知を無効にする」を選択します。

6. Modify 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、次のパラメータmodify-cache-clusterで 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、次のパラメータを使用して 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、クラスターインスタンス、セキュリティグループ、およびパラメータグループに関連するイベントをログに記録します。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、 コマンド、 AWS CLI `describe-events`または ElastiCache API アクション を使用して、ログからイベントを簡単に取得できます`DescribeEvents`。

次の手順では、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ます。

ElastiCache イベントの表示 (コンソール)

次の手順では、 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イベントを表示するには

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. 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 [イベント] を選択します。

   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 (キャッシュクラスター、 cache-security-group、または cache-subnet-group ） cache-parameter-group、イベントのGMT時刻、および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)

を使用して ElastiCache イベントのリストを生成するには AWS CLI、 コマンド を使用します`describe-events`。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)

を使用して ElastiCache イベントのリストを生成するには ElastiCache API、 DescribeEventsアクションを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードは、40 個の最新の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 は、キャッシュクラスターで重大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公開できます。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**
>
> 料金情報や Amazon SNSドキュメントへのリンクなど、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、指定された 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 通知には 1 つのトピックのみを設定できます。
- 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である必要があります。
- 公開先の Amazon SNSトピックは暗号化できません。

> **Note**
>
> 暗号化された (保管中の) 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して表示され、 がメッセージをトピックに ElastiCache プッシュすると、トピックとクラスターと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ある必要があります。

ElastiCache イベント

次の ElastiCache イベントは Amazon SNS通知をトリガーします。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 : cache-cluster`	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: 空き IP アドレスが不足AddCacheNodeFailedしているため	`ElastiCache:AddCacheNodeFailed : cluster-name`	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 : cluster-name`	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 cluster-name-0001-005`	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: 互換性のないネットワーク状態Cach	`ElastiCache:CacheClusterProvisioning`	存在しない仮想プライベートクラウド () に新しいキャッ

イベント名	メッセージ	説明
eClusterProvisioningFailed のため	`Failed : cluster-name`	シュクラスターを起動しようとしましたVPC。
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete : cluster-name`	キャッシュクラスターのスケーリングが正常に完了しました。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	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 : cluster-name`	以下のいずれかのイベントが発生しました。 • 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。 • キャッシュクラスターに関連付けられたキャッシュEC2セキュリティグループのいずれかで、1 つ以上の新しいセキュリティグループが承認されています。 • 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ているキャッシュセキュリティグループのいずれか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 : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストの劣化または到達不能を検出し、キャッシュノードの置き換えを開始しました。 **Note** 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。 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、 がキャッシュノードを ElastiCache 置き換えた後でも、キャッシュノードの使用を停止する場合があります。この場合、アプリケーションはこのイベントが発生したときにサーバーリストを更新する必要があります。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストの劣化または到達不能を検出し、キャッシュノードの交換を完了しました。 **Note** 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。 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、 がキャッシュノードを ElastiCache 置き換えた後でも、キャッシュノードの使用を停止する場合があります。この場合、アプリケーションはこのイベントが発生したときにサーバーリストを更新する必要があります。

イベント名	メッセージ	説明
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	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（バルキーまたは Redis OSSのみ）	`ElastiCache:CertificateRenewalComplete`	Amazon CA 証明書が正常に更新されました。
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete : cluster-name`	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 : cluster-name`	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（バルキーまたは Redis OSSのみ）	`ElastiCache:FailoverComplete : mycluster`	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 : cluster-name-0001-005`	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 : cluster-name-0003-004`	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 : cluster-name`	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 : cluster-name`	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、「ノードの交換 (Valkey と Redis OSS)」を参照してください。
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	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、「ノードの交換 (Valkey と Redis OSS)」を参照してください。
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	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。

イベント名	メッセージ	説明
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	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。
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（バルキーまたは Redis OSSのみ）	`ElastiCache:SnapshotComplete : cluster-name`	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（バルキーまたは Redis OSSのみ）	`SnapshotFailed : cluster-name`	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。

関連トピック

- ElastiCache イベントの表示

ログ配信

> **Note**
>
> スローログは、Valkey 7.x 以降、およびエンジンバージョン 6.0 以降を使用する Redis OSS キャッシュクラスターとレプリケーショングループでサポートされています。
> エンジンログは、Valkey 7.x、およびエンジンバージョン 6.2 以降の Redis OSSキャッシュクラスターとレプリケーショングループでサポートされています。

ログ配信を使用すると、次の 2 つの宛先のいずれかに SLOWLOGまたは エンジンログをストリーミングできます。

- Amazon Data Firehose
- Amazon CloudWatch Logs

を使用してクラスターを作成または変更するときに、ログ配信を有効にして設定します ElastiCache APIs。各ログエントリは、 JSONまたは 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TEXT。

固定数のスローログエントリがエンジンから定期的に取得されます。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 のいずれかの変更 を使用して、配信設定を変更するかAPIs、ログ配信を無効にするかを選択できますmodify-cache-clustermodify-replication-group。

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 の料金は、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」の「Vended Logs」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

スローログには次の情報が含まれます。

- 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 bar` fooはキー、 `bar` は value です。 は実際のキー名と値を ElastiCache に置き換え`(2 more arguments)`て、機密データを公開しないようにします。
- 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 エンジンログには、次の情報が含まれます。

- 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("-")`、 `NOTICE("*")`のいずれか LogLevel を使用できますWARNING("#")。
- 時刻 – ログに記録されたメッセージのUTC時刻。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。これは、プライマリの場合は「M」、レプリカの場合は「S」、センチネルの場合は RDB/AOF で作業するライターの子プロセスの場合は「C」のいずれかになります。
- メッセージ – エンジンログメッセージ。

ログ記録を設定するアクセス許可

IAM ユーザー/ロールポリシーには、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 の両方をサポートします TEXT

JSON フォーマット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 フォーマット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 の両方をサポートします TEXT

JSON フォーマットの例を次に示します。

```
{
  "CacheClusterId": "xxxxxxxxxzy-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 フォーマットの例を次に示します。

```
xxxxxxxxxxxzy-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、ログに選択できる ElastiCache ログ記録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。ログ記録先を設定したら、その仕様を ElastiCache ログ記録設定に提供して、ログ記録を開始できます。

トピック

- Amazon CloudWatch Logs
- Amazon Data Firehose

Amazon CloudWatch Logs

- CloudWatch ログを配信するロググループを指定します。
- 複数の Valkey または Redis OSSクラスターとレプリケーショングループのログは、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 ログに発行するアクセス許可

CloudWatch Logs ロググループにログ ElastiCache を送信するように 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 に送信された 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 Valkey または Redis OSSクラスターとレプリケーショングループのログは、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) リージョンでは利用できません。

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 ElastiCache ように 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

を使用して、 の手順に従って Valkey または Redis OSS (クラスターモードが無効) クラスター AWS Management Console を作成するValkey (クラスターモードが無効) クラスターの作成 (コンソール) か、 の手順に従って Valkey または Redis OSS (クラスターモードが有効) クラスターを作成できますValkey または Redis OSS (クラスターモードが有効) クラスターの作成 (コンソール) 。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. 詳細設定 で、ログ を選択し、スローログまたはエンジンログ のいずれかをチェックします。
2. ログ形式 で、テキストまたは を選択しますJSON。
3. 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。
4. Log destination で、Create new を選択し、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 を選択し、Logs CloudWatch 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、 で ElastiCache コンソールを開きますhttps://console.aws.amazon.com/elasticache/。
2. ナビゲーションペインから、Valkey クラスターまたは Redis OSS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。

    - 送信先タイプ で、CloudWatch ログ または Kinesis Firehose を選択します。

- Log destination で、Create new 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 ストリーム名のいずれかを選択します。

を使用したログ配信の指定 AWS CLI

スローログ

CloudWatch Logs へのログ配信が遅い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"
      }'
```

レプリケーショングループを変更して CloudWatch ログにスローログを配信する

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",
        "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"
      }'
```

Firehose にエンジンログを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 メトリクスの使用のモニタリング

ElastiCache には、クラスターをモニタリングできるメトリクスが用意されています。これらのメトリクスには、 からアクセスできます CloudWatch。の詳細については CloudWatch、 CloudWatch ドキュメントを参照してください。

ElastiCache は、ホストレベルのメトリクス (CPU使用状況など) 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 (キャッシュ取得やキャッシュミスなど) の両方を提供します。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ときに通知を受け取るように、特定のキーメトリクスに CloudWatch アラームを設定することを検討する必要があります。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

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。

- Valkey と Redis のメトリクス OSS

メトリクス	説明	単位
CPUUtilization	ホスト全体のCPU使用率。Valkey と Redis OSSはシングルスレッドであるため、4 つ以上の を持つノードのEngineCPUUtilization メトリクスをモニタリングすることをお勧めしますvCPUs。	割合 (%)
CPUCreditBalance	インスタンスが起動または開始されてから蓄積された獲得CPUクレジットの数。T2 Standard の場合、 には蓄積された起動クレジットの数CPUCreditBalanceも含まれます。 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 スタンダードの場合、起	クレジット (vCPU-分)

メトリクス	説明	単位
	動クレジットは制限に対してカウントされません。 のクレジットCPUCreditBalanceは、インスタンスがベースラインCPU使用率を超えてバーストするために使用できます。 CPU クレジットメトリクスは 5 分の頻度でのみ利用できます。 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。	
CPUCreditUsage	インスタンスがCPU使用率のために消費したCPUクレジットの数。1 つのCPUクレジットは、1 分間 100% 使用率で実行される 1 vCPU、または vCPUs、使用率、時間の同等の組み合わせ (例えば、2 分間 50% 使用率で実行される 1 vCPU、2 分間 25% 使用率で vCPUs 実行される 2 v) に相当します。 CPU クレジットメトリクスは 5 分の頻度でのみ利用できます。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。 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。	クレジット (vCPU-分)
FreeableMemory	ホストで使用可能な空きメモリの量。これは RAM、OS が解放可能とレポートする 、バッファ、キャッシュから派生します。	バイト
NetworkBytesIn	ホストがネットワークから読み取ったバイト数。	バイト

メトリクス	説明	単位
NetworkBytesOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたバイトの数。	バイト
NetworkPacketsIn	すべてのネットワークインターフェイスでの、このインスタンスによって受信されたパケットの数。このメトリクスは、受信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkPacketsOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたパケットの数。このメトリクスは、送信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkBandwidthInAllowanceExceeded	インバウンド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数。	バイト

メトリクス	説明	単位
NetworkMaxBytesOut	1 分あたりの送信バイトの 1 秒あたりの最大バースト。	バイト
NetworkMaxPacketsIn	1 分あたりのバースト受信パケットの最大数。	カウント
NetworkMaxPacketsOut	1 分あたりの送信パケットの 1 秒あたりの最大バースト数。	カウント
SwapUsage	ホストで使用されるスワップの量。	バイト

Valkey と Redis のメトリクス OSS

Amazon ElastiCache 名前空間には、次の Valkey メトリクスと Redis OSSメトリクスが含まれます。これらのメトリクスは、Valkey エンジンを使用する場合も同じです。

ReplicationLag と を除きEngineCPUUtilization、これらのメトリクスは info コマンドから派生します。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。

info コマンドの完全なドキュメントについては、http://valkey.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active_defrag_hits 、 の統計から派生しますINFO。	数
AuthenticationFailures	AUTH コマンドOSSを使用して Valkey または Redis に対して認証に失敗した試行の合計数。ACL LOG コマンドを使用して、個々の認証の失敗に関する詳細情報を確認できます。不正ア	カウント

メトリクス	説明	単位
	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。	
BytesUsedForCache	Valkey または Redis によって、データセット、バッファなど、OSSすべての目的に割り当てられたバイトの合計数。	バイト
	Dimension: Tier=Memory を使用する Valkey または Redis OSSクラスターの場合のデータ階層化 ElastiCache: メモリによるキャッシュに使用されるバイトの合計数。これは、のused_memory 統計の値ですINFO。	バイト
	Dimension: Tier=SSD を使用する Valkey または Redis OSSクラスターの場合のデータ階層化 ElastiCache: によるキャッシュに使用されるバイトの合計数SSD。	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。のデータ階層化 ElastiCache 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。のデータ階層化 ElastiCache を使用するクラスターのみがサポートされます。	バイト
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keyspace_hits 、の統計から派生しますINFO。	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keyspace_misses 、 の統計から派生しますINFO。	カウント

メトリクス	説明	単位
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。ACL LOG コマンドを使用して、個々の認証の失敗に関する詳細情報を確認できます。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	Valkey または Redis OSS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、cache_hits と cache_misses 統計を使用して、次の方法で計算されます: cache_hits /(cache_hits + cache_misses) 。	割合 (%)
ChannelAuthorizationFailures	ユーザーがアクセス許可を持たないチャネルへのアクセスに失敗した試行の合計数。ACL LOG コマンドを使用して、個々の認証の失敗に関する詳細情報を確認できます。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
CurrConnections	read replicas. からの接続を除くクライアント接続の数は、2 つから 4 つの接続 ElastiCache を使用して、各ケースでクラスターをモニタリングします。これは、 connected_clients の統計から派生しますINFO。	カウント
CurrItems	キャッシュの項目数。これはkeyspace統計から派生し、キースペース全体のすべてのキーを合計します。	カウント
	のデータ階層化 ElastiCache を使用するクラスターの Dimension: Tier=Memory です。メモリ内の項目の数です。	カウント

メトリクス	説明	単位
	のデータ階層化 ElastiCache を使用するクラスターの Dimension: Tier=SSD (ソリッドステートドライブ) です。内の項目の数SSD。	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expires統計から導き出され、すべてのキーをキースペース全体に設定された ttl で合計します。	カウント
DatabaseCapacityUsagePercentage	使用中のクラスターの総データ容量の割合。 Data Tiered 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) 、 used_memory と maxmemory は から取得されますINFO。 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 。	割合 (%)
DatabaseCapacityUsageCountedForEvictPercentage	オーバーヘッドと に使用されるメモリを除く、使用中のクラスターの合計データ容量の割合COB。このメトリクスは次のように計算されます。 used_memory - mem_not_counted_for_evict/maxmemory 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。 (used_memory + SSD used) / (maxmemory + SSD total capacity) used_memory および maxmemory が取得される場所 INFO	割合 (%)

メトリクス	説明	単位
DatabaseMemoryUsagePercentage	使用中のクラスターのメモリの割合。これは、used_memory/maxmemory から を使用して計算されますINFO。	割合 (%)
DatabaseMemoryUsageCountedForEvictPercentage	オーバーヘッドと 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COB。これは、 used_memory-mem_not_counted_for_evict/maxmemory から を使用して計算されますINFO。	割合 (%)
DB0AverageTTL	INFO コマンドの統計avg_ttlDBOからkeyspace 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期限切れにする (または のためにキーを削除するLRU) と、 DEL コマンドが合成され、すべてのレプリカに送信されます。したがって、レプリカノードの DB0AverageTTL は 0 です。これは、キーの有効期限が切れていないため、 を追跡しないためですTTL。	ミリ秒

メトリクス	説明	単位
EngineCPUUtilization	Valkey または Redis OSS エンジンスレッドのCPU使用率を提供します。Valkey と Redis OSSはシングルスレッドであるため、このメトリクスを使用してプロセス自体の負荷を分析できます。EngineCPUUtilization メトリクスは、プロセスのより正確な可視性を提供します。CPUUtilization メトリクスと組み合わせて使用できます。は、他のオペレーティングシステムや管理プロセスなど、サーバーインスタンス全体のCPU使用率をCPUUtilization 公開します。4 vCPUs つ以上の大規模なノードタイプでは、EngineCPUUtilization メトリクスを使用してスケーリングのしきい値をモニタリングおよび設定します。 **Note** 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ワークロードの大部分を占める可能性があります。これは、 が 3 つを超える大規模なホストでは重要ではありませんvCPUs。ただし、2 vCPUs 以下の小さいホストに影響す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、ホストが Valkey または Redis からの使用率が高くOSS、バックグラウンドモニタリングプロセスCPUからのCPU使用率が高いという状況に気付かないでしょう。したがって、2 vCPUsつ以下	割合 (%)

メトリクス	説明	単位
	のホストのCPUUtilization メトリクスをモニタリングすることをお勧めします。	
Evictions	maxmemory の制限のため排除されたキーの数。これはevicted_keys 、 の統計から派生しますINFO。	カウント
GlobalDatastoreReplicationLag	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になっている Valkey または Redis の場合 OSS、遅延はシャード間の最大遅延を示します。	[秒]
IamAuthenticationExpirations	期限切れのIAM認証済み Valkey または Redis OSS接続の合計数。IAM による認証 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認証済み Valkey または Redis OSSAUTHまたはHELLOリクエストの総数。IAM による認証 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
IsMaster	ノードが現在のシャード/クラスタのプライマリノードかどうかを示します。メトリクスは 0 (プライマリではない) または 1 (プライマリ) にすることができます。	カウント
KeyAuthorizationFailures	ユーザーがアクセス許可を持たないキーへのアクセスに失敗した試行の合計数。ACL LOG コマンドを使用して、個々の認証の失敗に関する詳細情報を確認できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント

メトリクス	説明	単位
KeysTracked	Valkey または Redis キー追跡によって追跡されるOSSキーの数を、 の割合で示しますtracking-table-max-keys 。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	Valkey または Redis OSS エンジンのメモリ割り当ての効率を示します。特定のしきい値は、異なる動作を意味します。推奨値は、1.0 を超える断片化です。これは、 mem_fragmentation_ratio statistic の から計算されますINFO。	数
NewConnections	この期間内にサーバーによって受け入れられた接続の総数。これは、 total_connections_received の統計から派生しますINFO。 **Note** ElastiCache (Redis OSS) バージョン 5 以前を使用している場合、このメトリクスによって報告された接続のうち 2 つから 4 つが 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、 ElastiCache (Redis OSS) バージョン 6 以降を使用する場合、クラスターのモニタリング ElastiCache に が使用する接続はこのメトリクスに含まれません。	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。のデータ階層化 ElastiCache 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

メトリクス	説明	単位
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。のデータ階層化 ElastiCache 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 OSSの Valkey または Redis と同期していないことを示しますEC2。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、 CompleteMigrationAPIオペレーションを使用します。	ブール値
Reclaimed	キーの有効期限切れイベントの総数。これは、 expired_keys の統計から派生しますINFO。	カウント
ReplicationBytes	レプリケートされたノードについては、 ReplicationBytes は、プライマリがすべてのレプリカに対して送信するバイト数を報告します。このメトリクスは、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master_repl_offset 、 の統計から派生しますINFO。	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。Valkey 7.2 以降、および Redis OSS エンジンバージョン 5.0.6 以降では、遅延をミリ秒単位で測定できます。	[秒]

メトリクス	説明	単位
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rdb_bgsave_in_progress 、 の統計から派生しますINFO。	ブール値

メトリクス	説明	単位
TrafficManagementActive	ElastiCache (Redis OSS) 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して、トラフィックを積極的に管理しているかどうかを示します。トラフィックは、Valkey または Redis が処理できるコマンドよりも多くのコマンドがノードに送信されると管理OSSされ、エンジンの安定性と最適なオペレーション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。 **Note** 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。	ブール値

E ngineCPUUtilization の可用性

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。コマンド統計セクションには、呼び出しの数、これらのコマンドが消費した合計CPU時間、コマンド実行あたりの平均CPU消費時間など、コマンドタイプに基づく統計が表示されます。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`。

次に示すレイテンシーメトリクスは、 のコマンド統計を使用して計算されますINFO。それらは次のように計算されます: `delta(usec)/delta(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、 コマンドの処理 ElastiCache に がかかった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、 から項目を取得するのにかかる時間はこれらの測定値に含まれSSDないことに注意してください。

使用可能なコマンドの完全なリストについては、「Valkey ドキュメント」の「 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 (`cluster slot`、 `cluster info`など) で動作するすべてのコマンドを合計することでcommandstats 統計から導き出されます。	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commandstats 、eval、 を合計して統計から派生しますevalsha。	カウント

メトリクス	説明	単位
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commandstats 統計から派生します。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
GeoSpatialBasedCmdsLatency	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型コマンド (get、hget、、lrangeなど) scardを合計してcommandstats 統計から派生します。	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 (hget、hkeys、、 hdelなど) hvalsで動作するすべてのコマンドを合計することでcommandstats 統計から派生します。	カウント
HashBasedCmdsLatency	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タイプのコマンド (pfadd、pfcount、 pfmergeなど) を合計してcommandstats 統計から派生します。	カウント
HyperLogLogBasedCmdsLatency	HyperLogLog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
JsonBasedCmds	読み取りJSONコマンドと書き込みコマンドの両方を含むコマンドの合計数。これは、JSONキーで動作するすべてのJSONコマンドを合計することでcommandstats 統計から導き出されます。	カウント
JsonBasedCmdsLatency	読み取りJSONコマンドと書き込みコマンドの両方を含む、すべての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、JSONキーで動作するすべてのJSON読み取りコマンドを合計することでcommandstats 統計から導き出されます。	カウント
JsonBasedGetCmdsLatency	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、JSONキーで動作するすべてのJSON書き込みコマンドを合計することでcommandstats 統計から導き出されます。	カウント
JsonBasedSetCmdsLatency	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 (del、、 expire renameなど) にわたって 1 つ以上のキーで動作するすべてのコマンドを合計することで、commandstats 統計から派生します。	カウント
KeyBasedCmdsLatency	キー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
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 (lindex、lrange、、 ltrimなど) に基づいて動作するすべてのコマンドを合計することでlpush、commandstats 統計から派生します。	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、 acldbsizeや など、キーに作用しないすべてのコマンドを合計することでcommandstats 統計から導き出されますinfo。	カウント
NonKeyTypeCmdsLatency	コマンドの non-key-based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 機能に使用されるすべてのコマンド、psubscribe、、publish、、pubsubpunsubscribe、ssubscribesunsubscribe、spublishsubscribe、および を合計してcommandstats 統計から派生しますunsubscribe。	カウント
PubSubBasedCmdsLatency	PubSubBased コマンドのレイテンシー。	マイクロ秒
SetBasedCmds	セットベースのコマンドの総数。これは、1 つ以上のセット (scard、sdiff、、 sunionなど) で動作するすべてのコマンドを合計commandstats することでsadd統計から派生します。	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

メトリクス	説明	単位
SetTypeCmds	write 型のコマンドの合計数。これは、データ (set、、hset、 sadd lpopなど) で動作するすべてのmutativeタイプのコマンドを合計することで、commandstats 統計から導き出されます。	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 (zcount、zrange、zadd、 など) で動作するすべてのコマンドを合計commandstats することでzrank統計から派生します。	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 (strlen、setex、 setrangeなど) に作用するすべてのコマンドを合計することでcommandstats 統計から導き出されます。	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
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型 (xrange、xlen、xadd、 xdelなど) に作用するすべてのコマンドを合計することでcommandstats 統計から導き出されます。	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

Memcached のメトリクス

AWS/ElastiCache 名前空間には、次の Memcached メトリクスが含まれています。

AWS/ElastiCache namespace には、Memcached stats コマンドから派生した以下のメトリクスが含まれます。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。

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
BytesReadIntoMemcached	キャッシュノードによってネットワークから読み取られたバイト数。	バイト
BytesUsedForCacheItems	キャッシュ項目の格納に使用したバイト数。	バイト
BytesWrittenOutFromMemcached	キャッシュノードによってネットワークに書き込まれたバイト数。	バイト
CasBadval	Cas 値が保存された Cas 値と一致しなかったキャッシュが受信した CAS (チェックして設定する) リクエストの数。	カウント
CasHits	キャッシュが受信し、リクエストされたキーが見つかって Cas 値が一致した Cas リクエストの数。	カウント
CasMisses	キャッシュが受信したが、リクエストされたキーが見つからない Cas リクエストの数。	カウント
CmdFlush	キャッシュが受信した flush コマンドの数。	カウント
CmdGet	キャッシュが受信した get コマンドの数。	カウント
CmdSet	キャッシュが受信した set コマンドの数。	カウント

メトリクス	説明	単位
CurrConnections	特定の時点でキャッシュに接続された接続回数。ElastiCache は 2 ~ 3 個の接続を使用してクラスターをモニタリングします。 上記に加えて、memcached は、ノードタイプに使用されているスレッドの 2 倍に等しい数の内部接続を作成します。ノードタイプ別のスレッド数は、該当するパラメータグループの Nodetype Specific Parameters で確認できます。 合計接続数は、クライアント接続、モニタリング用の接続、および上記の内部接続の合計数です。	カウント
CurrItems	キャッシュに現在格納されている項目の数。	カウント
DecrHits	キャッシュが受信し、リクエストされたキーが見つかったデクリメントリクエストの数。	カウント
DecrMisses	キャッシュが受信したが、リクエストされたキーが見つからなかったデクリメントリクエストの数。	カウント
DeleteHits	キャッシュが受信し、リクエストされたキーが見つかった削除リクエストの数。	カウント
DeleteMisses	キャッシュが受信したが、リクエストされたキーが見つからなかった削除リクエストの数。	カウント
Evictions	新しく書き込むための領域を確保するためにキャッシュが排除した、期限切れではない項目の数。	カウント
GetHits	キャッシュが受信し、リクエストされたキーが見つかった get リクエストの数。	カウント

メトリクス	説明	単位
GetMisses	キャッシュが受信したが、リクエストされたキーが見つからなかった get リクエストの数。	カウント
IncrHits	キャッシュが受信し、リクエストされたキーが見つかったインクリメントリクエストの数。	カウント
IncrMisses	キャッシュが受信したが、リクエストされたキーが見つからなかったインクリメントリクエストの数。	カウント
Reclaimed	新しく書き込むための領域を確保するためにキャッシュが排除した、期限切れ項目の数。	カウント

Memcached 1.4.14 では、次のメトリクスが追加で提供されます。

メトリクス	説明	単位
BytesUsedForHash	ハッシュテーブルで現在使用されているバイト数。	バイト
CmdConfigGet	config get リクエストの累積数。	カウント
CmdConfigSet	config set リクエストの累積数。	カウント
CmdTouch	touch リクエストの累積数。	カウント
CurrConfig	現在格納されている設定の数。	カウント
EvictedUnfetched	最後に使用したキャッシュ (LRU) から取り出した有効な項目のうち、設定後に一度も触れなかったものの数。	カウント
ExpiredUnfetched	設定後に一度も触れなかった から再利用された期限切れLRUのアイテムの数。	カウント
SlabsMoved	移動されたスラブページの合計数。	カウント

メトリクス	説明	単位
TouchHits	タッチされて新しい有効期限を与えられたキーの数。	カウント
TouchMisses	タッチされたが見つからなかった項目の数。	カウント

AWS/ElastiCache namespace には、以下の計算されたキャッシュレベルのメトリクスが含まれます。

メトリクス	説明	単位
NewConnections	キャッシュが受信した新しい接続の数。この値は、ある期間の変更を total_connections に記録することによって memcached total_connections 統計から派生したものです。これは、用に予約された接続のため、常に 1 以上になります ElastiCache。	カウント
NewItems	キャッシュが格納した新しい項目の数。この値は、ある期間の変更を total_items に記録することによって memcached total_items 統計から派生したものです。	カウント
UnusedMemory	データに使用されていないメモリの量。この値は、Memcached 統計の limit_maxbytes と bytes の間で、limit_maxbytes から bytes を引くことによって算出されます。 Memcached オーバーヘッドは、データで使用されるメモリに加えてメモリを使用するため、追加データに使用できるメモリ量と見な UnusedMemory すべきではありません。未使用メモリがまだ残っている状態でも削除が発生する場合があります。	バイト

メトリクス	説明	単位
	詳細については、「Memcached item memory usage」を参照してください。	

モニタリングすべきメトリクス

以下の CloudWatch メトリクスは、 ElastiCache パフォーマンスに関する優れたインサイトを提供します。ほとんどの場合、パフォーマンスの問題が発生する前に是正措置を講じることができるように、これらのメトリクスの 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 （ バルキーと Redis OSS ）
- Evictions
- CurrConnections
- メモリ (Valkey と Redis OSS)
- 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 (バルキーと Redis OSS)

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。

Valkey と Redis OSS

2 vCPUs 以下の小さいノードタイプでは、 `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。

一般的には、しきい値を使用可能な の 90% に設定することをお勧めしますCPU。Valkey と Redis OSSはどちらも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合計容量の分数として計算する必要があります。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、 のしきい値は 90/2 または 45% CPUUtilization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。

- Valkey または Redis OSS (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。
- Valkey または Redis OSS (クラスターモードが有効) クラスター: さらにシャードを追加して、書き込みワークロードをより多くのプライマリノードに分散します。

> **Tip**
>
> ホストレベルのメトリクス を使用する代わりにCPUUtilization、Valkey および Redis OSSユーザーはメトリクス を使用できる場合があります。メトリクスはEngineCPUUtilization、Valkey または Redis OSS エンジンコアの使用率をレポートします。このメトリクスがノードで使用可能かどうかを確認し、詳細については、「Valkey と Redis のメトリクスOSS」を参照してください。

4 vCPUs 個以上の大きなノードタイプでは、Valkey または Redis OSS エンジンコアの使用率をレポートするEngineCPUUtilizationメトリクスを使用できます。このメトリクスがノードで使用可能かどうかを確認し、詳細については、「Redis のメトリクスOSS」を参照してください。

Memcached

Memcached はマルチスレッドのため、このメトリクスは約 90% です。このしきい値を超えた場合は、より大きなキャッシュノードタイプを使用してキャッシュクラスターをスケールアップするか、キャッシュノードをさらに追加してスケールアウトします。

EngineCPUUtilization

4 vCPUs つ以上のより大きなノードタイプでは、Redis OSS エンジンコアの使用率をレポートするEngineCPUUtilizationメトリクスを使用できます。このメトリクスがノードで使用可能かどうかを確認し、詳細については、「Valkey と Redis のメトリクスOSS」を参照してください。

詳細については、「Amazon を使用した Amazon ElastiCache (Redis OSS) でのベストプラクティスのモニタリング CloudWatchCPUs」セクションを参照してください。

SwapUsage（バルキーと Redis OSS）

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 (1100MB) か、SwapUsageメトリクスがFreeableMemoryメトリクスより大きい場合は、ノードがメモリ圧力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。

- Valkey または Redis OSS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を確認する
- Valkey と Redis の予約済みメモリの管理 OSS

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。

Memcached を使用していて、選択したしきい値を超える場合は、より大きなノードタイプを使用してクラスターをスケールアップするか、ノードを追加してスケールアウトします。

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。

の数が増えると、アプリケーションに問題があるCurrConnections可能性があります。この問題に対処するには、アプリケーションの動作を調べる必要があります。

詳細については、「Amazon を使用した Amazon (Redis) でのベストプラクティスのモニタリング」の「Connections」セクションを参照してください。 ElastiCache OSS CloudWatch

メモリ (Valkey と Redis OSS)

メモリは Valkey と Redis の中核的な側面ですOSS。クラスターのメモリ使用率を理解することは、データの損失を回避し、データセットの将来の増加に対応するために必要です。ノードのメモリ使用率に関する統計は、 INFO コマンドのメモリセクションで確認できます。

詳細については、「Amazon を使用した Amazon (Redis) での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。 ElastiCache OSS CloudWatch

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 の ElastiCache 料金」を参照してください。

詳細については、「Amazon を使用した Amazon ElastiCache (Redis OSS) でのベストプラクティスのモニタリング CloudWatch」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

コマンドのレイテンシーは、データ構造ごとに集約されたレイテンシーを提供する一連の CloudWatch メトリクスを使用して測定できます。これらのレイテンシーメトリクスは、Valkey INFO コマンドのcommandstats統計を使用して計算されます。

詳細については、「Amazon を使用した Amazon ElastiCache でのベストプラクティスのモニタリング CloudWatch」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 を使用した Amazon ElastiCache (Redis OSS) でのベストプラクティスのモニタリング CloudWatch」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 (バルキーと Redis OSS)

ElastiCache (Redis OSS) は、Valkey または Redis で処理できる数よりも多くの受信コマンドがノードに送信されると、ノードに対するトラフィックを自動的に管理しますOSS。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、メトリクスごとに任意の統計と期間を選択できますが、すべての組み合わせが役立つわけではありません。例えば、 の平均統計、最小統計、最大統計CPUUtilizationは便利ですが、合計統計は有用ではありません。

すべての ElastiCache サンプルは、個々のキャッシュノードごとに 60 秒間発行されます。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められるサンプルは 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の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のメトリクスのモニタリング

ElastiCache と CloudWatch は統合されているため、さまざまなメトリクスを収集できます。これらのメトリクスは、 を使用してモニタリングできます CloudWatch。

> **Note**
>
> 次の例では、 CloudWatch コマンドラインツールが必要です。 CloudWatch および デベロッパーツールの詳細については、 CloudWatch 製品ページ を参照してください。

次の手順では、 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スペース統計を収集する方法を示します。

> **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、AWS 「 のサービス制限」を参照してください ElastiCache。

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. 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. 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キャッシュクラスターページで、1 つ以上のキャッシュクラスターの名前をクリックします。

   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 使用率メトリクスをクリックします。

   CloudWatch コンソールが開き、選択し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。

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 CloudWatch CLI

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
```

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 CloudWatch API

キャッシュクラスターのCPU使用率統計を収集するには

- 次のパラメータGetMetricStatisticsを使用して を CloudWatch API呼び出します (開始時刻と終了時刻は例としてのみ表示されることに注意してください。独自の適切な開始時刻と終了時刻を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>
```

のクォータ ElastiCache

AWS アカウントには、 AWS サービスごとに、以前は制限と呼ばれていたデフォルトのクォータがあります。特に明記されていない限り、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。

のクォータを表示するには ElastiCache、Service Quotas コンソール を開きます。ナビゲーションペインで、AWS サービスを選択し、 を選択しますElastiCache。

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、[上限引き上げ] フォームを使用してください。

AWS アカウントには、 に関連する次のクォータがあります ElastiCache。

リソース	デフォルト
リージョンごとのサーバーレスキャッシュ	40
キャッシュあたりの 1 日あたりのサーバーレススナップショット、Redis	24
リージョンあたりのノード	300
クラスターあたりのノード数、Memcached	60
インスタンスタイプ、Valkey または Redis あたりのクラスターあたりのノードOSS数 (クラスターモードが有効)	90
シャード、バルキー、または Redis あたりのノードOSS数 (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

リソース	デフォルト
ユーザーグループあたりのユーザー、Redis	100
最大ユーザー数、Redis	1,000
ユーザーグループの最大数、Redis	100

リファレンス

このセクションのトピックでは、Amazon ElastiCache API と ElastiCache のセクションの操作について説明します AWS CLI。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。

- の使用 ElastiCache API
- ElastiCache API リファレンス
- ElastiCache AWS CLI リファレンスの セクション
- Amazon ElastiCache エラーメッセージ
- 通知

の使用 ElastiCache API

このセクションでは、 ElastiCache オペレーションの使用と実装に関するタスク指向の説明を提供します。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

- クエリの使用 API
- 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

クエリの使用 API

クエリパラメータ

HTTP クエリベースのリクエストはHTTP、動GET詞または POST と という名前のクエリパラメータを使用するHTTPリクエストです`Action`。

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。の値 *`n`* は 1 から始まる整数です。

クエリリクエストの認証

クエリリクエストは 経由でのみ送信できHTTPS、すべてのクエリリクエストに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。これは、 に登録 AWS 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があ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 へのリクエストを構築します AWS。
2. 送信者は、このトピックの次のセクションで定義されているように、SHA-1 ハッシュ関数を使用して、リクエスト署名であるハッシュベースのメッセージ認証コード (HMAC) のキードハッシュを計算します。
3. リクエストの送信者は、リクエストデータ、署名、アクセスキー ID (使用されているシークレットアクセスキーのキー識別子) を に送信します AWS。
4. AWS 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検索します。
5. AWS は、リクエスト内の署名の計算に使用されるのと同じアルゴリズムを使用して、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. UTF-8 クエリ文字列コンポーネントをパラメータ名で自然バイト順序でソートします。パラメータは、 GET URI またはPOST本文 (Content-Type が application/ の場合x-www-form-urlencoded) から取得できます。

b. URL 次のルールに従って、パラメータ名と値をエンコードします。

   i. RFC 3986 URL が定義する予約されていない文字をエンコードしないでください。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. %XY%ZA の形式で拡張 UTF-8 文字をエンコードするパーセント。

   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. パラメータ値が空であっても、エンコードされたパラメータ名をエンコードされた値から等号 (=) (ASCII 文字 61) で区切ります。

d. 名前と値のペアをアンパサンド (&) (ASCII コード 38) で区切ります。

2. 次の擬似文法に従って署名する文字列を作成します (「\n」はASCII改行を表します) 。

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

   HTTPRequestURI コンポーネントは、クエリ文字列URIまでのHTTP絶対パスコンポーネントです。HTTPRequestURI が空の場合は、スラッシュ (/) を使用します。

3. 先ほど作成した文字列HMAC、キーとしてのシークレットアクセスキー、ハッシュアルゴリズムSHA256SHA1として RFC2104 準拠を計算します。

   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
```

上記のクエリ文字列では、次の文字列でHMAC署名を計算します。

```
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 は、 クエリ APIs の代わりに言語固有のアプリケーションを使用してアプリケーションを構築するソフトウェアデベロッパー向けのソフトウェア開発キット (SDKs) を提供しますAPI。これらは、リクエスト認証、リクエスト再試行、エラー処理などの基本的な機能 (に含まれていないAPIs) SDKsを提供し、開始が容易になります。SDKs および の追加のリソースは、次のプログラミング言語で使用できます。

- Java
- Windows と 。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 には、 の操作中に問題をトラブルシューティングするのに役立つ具体的で説明的なエラーが用意されています ElastiCache API。

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、 からのレスポンスで`Error`ノードを探すことです ElastiCache API。

XPath 構文は、`Error`ノードの存在を検索する簡単な方法と、エラーコードとメッセージを取得する簡単な方法を提供します。次のコードスニペットは、Perl と XML::XPath モジュールを使用して、リクエスト中にエラーが発生したかどうかを判断します。エラーが発生した場合、レスポンス内の最初のエラーコードとメッセージが表示されます。

```
use XML::XPath;
```

```
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

トラブルシューティングのヒント

に関する問題を診断して解決するには、以下のプロセスをお勧めします ElastiCache API。

- 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き、 ElastiCache サービス (など https://elasticache.amazonaws.com) にクエリリクエストを送信します。MissingAuthenticationTokenException または 500 Internal Server Error は、サービスが利用可能であり、リクエストに応答することを確認します。

- リクエストの構文を確認します。

  各 ElastiCache オペレーションには、リファレンス ElastiCache API に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、このフォーラムでは、その過程で他の人が経験した問題の解決策を検索できます。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セットアップ

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。

このトピックの手順に従うのは、 AWS CLI の にアクセスする場合のみです ElastiCache。

> **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 改善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

このドキュメントは、Linux/UNIXまたは Windows 環境で作業できることを前提としています。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 でも機能しますが、このガイドには特定の Mac OS X 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**$** 、 を使用して Linux/UNIX特定のコマンドを指定し、Windows 固有のコマンド**C:\>** を指定します。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のインストールは許容されます。Linux/UNIX および Windows を含むJREsさまざまなプラットフォームの表示とダウンロードについては、「Java SE ダウンロード」を参照してください。

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、 という名前のサブディレクトリを含むディレクトリのフルパスに設定す

る必要があります。binサブディレクトリには、実行可能ファイル java (Linux および UNIX) または java.exe (Windows) 実行可能ファイルが含まれます。

JAVA_HOME 変数の設定方法

1. Java ホーム変数を設定します。

   - Linux と でUNIX、次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/ ではUNIX、次のような出力が表示されます。

     ```
     $ $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。

     ```
     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 デベロッパーツールウェブサイト でZIPファイルとして利用できます。これらのツールは Java で記述され、Windows 2000/XP/Vista/Windows 7、Linux/UNIX、Mac のシェルスクリプトが含まれていますOSX。ZIP ファイルは自己完結型で、インストー

ルは必要ありません。zip ファイルをダウンロードして、ローカルマシンのディレクトリに解凍します。

ツールを設定する

コマンドラインツールは、環境変数 (AWS_ELASTICACHE_HOME) に依存して、サポートされているライブラリを見つけます。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、n はバージョン/リリース番号) で、bin と lib という名前のサブディレクトリが含まれています。

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 でUNIX、次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>
```

ツールをより使いやすくするために、ツールのBINディレクトリをシステム に追加することをお勧めしますPATH。このガイドの残りの部分では、BINディレクトリがシステムパスにあ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

- 次のコマンドを入力して、ツールのBINディレクトリをシステム に追加しますPATH。
  - Linux と でUNIX、次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。

デプロイには、情報で編集する必要がある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。

```
AWS AccessKeyId=<Write your AWS access ID>
AWS SecretKey=<Write your AWS secret key>
```

> **Important**
>
> ではUNIX、認証情報ファイルの所有者にアクセス許可を制限します。
>
> ```
> $ chmod 600 <the file created above>
> ```

認証情報ファイルのセットアップでは、 ElastiCache ツールが情報を検索できるように AWS_CREDENTIAL_FILE 環境変数を設定する必要があります。

AWS_CREDENTIAL_FILE 環境変数を設定するには

1. 環境変数を設定します:

   - Linux および ではUNIX、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コマンドの使用状況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含まれるほとんどの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。
- AWS_ELASTICACHE_URL — サービスコールURLに使用する 。EC2_REGION が指定されている場合、または --region パラメータが渡されている場合、別のリージョンエンドポイントを指定する必要はありません。

次の例は、環境変数 EC2_REGION を設定して、APIツールで使用されるリージョンを設定する方法を示しています。

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

次のエラーメッセージは Amazon によって返されます ElastiCache。、他の AWS サービス、または Valkey ElastiCache、Redis、OSSまたは Memcached によって返される他のエラーメッセージが表示される場合があります。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 ElastiCache、エラーメッセージを生成しているソース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 を超えるノードが必要な場合は、Amazon ElastiCache Node リクエストフォーム を使用してリクエストを行います。

詳細については、「」の「Amazon ElastiCache Limits」を参照してくださいAmazon Web Services 全般のリファレンス。

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 を超えるノードが必要な場合は、Amazon ElastiCache Node リクエストフォーム を使用してリクエストを行います。

詳細については、「」の「Amazon ElastiCache Limits」を参照してくださいAmazon Web Services 全般のリファレンス。

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ました。

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、「スケーリング ElastiCache」を参照してください。

通知

このトピックでは、関心がある可能性のある ElastiCache 通知について説明します。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、新機能や改善された ElastiCache 機能を発表しません。

一般的な ElastiCache 通知

現在、エンジン固有ではない未処理 ElastiCache の通知はありません。

ElastiCache (Memcached) 通知

次の ElastiCache 通知は、Memcached エンジンに固有のものです。

ElastiCache (Memcached) 固有の通知

- アラート: セグメンテーション障害を引き起こす Memcached LRU クローラ

アラート: セグメンテーション障害を引き起こす Memcached LRU クローラ

アラート日付: 2017 年 2 月 28 日
状況によっては、クラスターが不安定になり、Memcached LRU クローラーにセグメンテーション障害が発生することがあります。これはしばらくの間存在している Memcached エンジン内の問題です。この問題は、クローラーがデフォルトで有効になっているときに Memcached 1.4.33 LRU で明らかになりました。
この問題が発生した場合は、修正されるまでLRUクローラーを無効にすることをお勧めします。そのためには、コマンドラインで `lru_crawler disable` を使用するか、または `lru_crawler` パラメータ値を変更します (推奨)。
解決した日付:
解決策:

ElastiCache (Redis OSS) 固有の通知

現在、未処理の ElastiCache (Redis OSS) 通知はありません。

ElastiCache 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 ユーザーガイドの各リリースにおける重要な変更点を示しています。このドキュメントの更新に関する通知については、RSSフィードをサブスクライブできます。

最近の ElastiCache 更新

変更	説明	日付
Valkey ElastiCache での のサポート	ElastiCache が Valkey をサポートするようになりました。Valkey 7.2.6 は Redis 7.2 OSS と互換性があります。詳細については、「 Valkey 」を参照してください。	2024 年 10 月 8 日
柔軟なリザーブドノードのサイズ	ElastiCache は、サイズフレキシブル予約ノード をサポートするようになりました。詳細については、「Amazon ElastiCache 料金」を参照してください。	2024 年 10 月 1 日
ElastiCache (Redis OSS) が追加の C7gn ノードサイズのサポートを追加しました	ElastiCache (Redis OSS) は、追加の C7gn ノードサイズのサポートを追加しました。	2024 年 1 月 10 日
ElastiCache (Redis OSS) がサーバーレスキャッシュの作成をサポートするようになりました	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	2023 年 11 月 27 日

	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とマネージドVPCエンドポイントの関連付けを許可するAmazonElastiCacheFullAccess 新しいアクセス許可が ElastiCacheServiceRolePolicy および 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 (Memcached) がサーバーレスキャッシュの作成をサポートするようになりました	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とマネージドVPCエンドポイントの関連付けを許可するAmazonElastiCacheFullAccess 新しいアクセス許可が ElastiCacheServiceRolePolicy および 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。	2023 年 11 月 27 日
ElastiCache (Redis OSS) がクラスターモードの変更をサポートするようになりました	クラスターをクラスターモード無効 (CMD) からクラスターモード有効 () に移行できるようになりましたCME。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 (Redis OSS) が転送中の暗号化設定の変更をサポートするようになりました	OSS 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TLS の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 (Redis OSS) は、を使用したユーザーの認証をサポートするようになりました。 IAM	IAM 認証を使用すると、アイデンティティを使用して ElastiCache AWS IAM (Redis OSS) への接続を認証できます。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。詳細については、「 による認証IAM」を参照してください。	2022 年 11 月 16 日
ElastiCache (Redis OSS) が Redis 7 OSS をサポートするようになりました	このリリースでは、Amazon ElastiCache (Redis OSS) に Redis OSS関数、ACL改善点、およびシャーデッド Pub/Sub といういくつかの新機能が導入されています。詳細については、 ElastiCache (Redis OSS) バージョン 7.0 を参照してください。	2022 年 11 月 8 日

ElastiCache (Redis OSS) がをサポートするようになりました IPV6	ElastiCache では、インターネットプロトコルバージョン 4 と 6 (IPv4 と IPv6) がサポートされているため、IPv4接続のみ、IPv6接続のみ、またはその両方 IPv4 IPv6 (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 OSS エンジンバージョン 6.2 以降のワークロードでサポートされています。 ElastiCache 経由で にアクセスするための追加料金は発生しませんIPv6。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 (Memcached) がをサポートするようになりました IPV6	ElastiCache では、インターネットプロトコルバージョン 4 と 6 (IPv4 と IPv6) がサポートされているため、IPv4接続のみ、IPv6接続のみ、またはその両方 IPv4 IPv6 (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、Memcached エンジンバージョン 1.6.6 以降のワークロードでサポートされています。 ElastiCache 経由で にアクセスするための追加料金は発生しませんIPv6。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 (Memcached) が転送中の暗号化をサポートするようになりました	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。Memcached バージョン 1.6.12 以降でサポートされています。詳細については、ElastiCache 「転送中の暗号化 (TLS) 」を参照してください。	2022 年 5 月 26 日

ElastiCache (Redis OSS) がネイティブ JavaScript Object Notation (JSON) 形式をサポートするようになりました	ネイティブ JavaScript の Object Notation (JSON) 形式は、Redis OSSクラスター内の複雑なデータセットをエンコードするシンプルでスキーマレスな方法です。Redis OSS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。シリアル化とシリアル化解除を行う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、「 の開始方法JSON」を参照してください。	2022 年 5 月 25 日
ElastiCache が をサポートするようになりました PrivateLink	AWS PrivateLink では、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、オペレーションにプライベートにアクセスできます ElastiCache API。詳細については、「Redis の場合は Amazon ElastiCache API およびインターフェイスVPCエンドポイント (AWS PrivateLink) OSS」、Memcached の場合は Amazon ElastiCache API およびインターフェイス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 (Redis OSS) が Redis OSS 6.2 とデータ階層化をサポートするようになりました	Amazon ElastiCache (Redis OSS) は、Amazon でサポートされている Redis OSS エンジンの次のバージョンを導入します ElastiCache。 ElastiCache（ Redis OSS) 6.2 には、8 vCPUs つ以上の x86 ノードタイプ、または 4 vCPUs つ以上の Graviton2 ノードタイプを使用する TLS対応クラスターのパフォーマンス向上が含まれています。ElastiCache (Redis OSS) 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 (Redis OSS) バージョン 6.2 (拡張) とデータ階層化 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 (Redis OSS) が Auto Scaling をサポートするようになりました。 ElastiCache （ Redis OSS) 自動スケーリングは、 ElastiCache (Redis OSS) サービスで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 (Redis OSS) クラスター」を参照してください。	2021 年 8 月 19 日
Redis Slow OSS ログの配信のサポート	ElastiCache では、Redis を Amazon Data Firehose または Amazon CloudWatch Logs の 2 つの宛先のいずれかOSSSLOWLOG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 では、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 リソースのタグ付け」を参照してください。 は、条件キーのサポート ElastiCache も導入しています。IAM ポリシーを有効にする方法を決定す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
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 では、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 リソースのタグ付け」を参照してください。 は、条件キーのサポート ElastiCache も導入しています。IAM ポリシーを有効にする方法を決定す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、ほぼすべてのデータセンター、コロケーションスペース、オンプレミス施設にネイティブ AWS サービス、インフラストラクチャ、運用モデルを提供します。Outposts ElastiCache にデプロイして、クラウドと同じよう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、「Outposts for Redis の使用OSS」または「Outposts for Memcached の使用」を参照してください。	2020 年 10 月 8 日
ElastiCache Redis 6 OSS をサポートするようになりました	Amazon ElastiCache (Redis OSS) は、Amazon でサポートされている Redis OSS エンジンの次のバージョンを導入します 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 (Redis OSS) 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 がローカルゾーンをサポートするようになりました	ローカルゾーンは、地理的にユーザーに近い AWS リージョンの拡張です。新しいサブネットを作成してローカルゾーンに割り当てることで、仮想プライベートクラウド (VPC) を親 AWS リージョンからローカルゾーンに拡張できます。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 (Redis OSS) は、Redis OSS クラスター環境を最大 500 ノードまたは 500 シャードにスケーリングできるようになりました。	Redis OSSクラスターモードでは、複数のシャードに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、可用性が向上します。この機能は、Amazon ElastiCache (Redis OSS) バージョン 5.0.6 以降では、すべての AWS リージョンと、既存のクラスター環境と新しいクラスター環境 ElastiCache (Redis OSS) で使用できます。詳細については、「Redis OSS Nodes and Shards」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 (Redis OSS) が Amazon CloudWatch メトリクスを追加	ElastiCache (Redis OSS) は、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OSS」を参照してください。	2020 年 6 月 10 日
ElastiCache が ElastiCache クラスターの自動更新をサポートするようになりました	Amazon は、「推奨される適用日」のサービス更新が経過した後、 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 (Redis OSS) が Global Datastore for Redis をサポートするようになりました OSS	Global Datastore for Redis OSS機能は、 AWS リージョン間でフルマネージド、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 (Redis OSS) 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。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。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、 AWS リージョンをグローバルデータストア内の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、「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。	2020 年 3 月 16 日
ElastiCache (Redis OSS) が Redis OSSバージョン 5.0.6 をサポートするようになりました	詳細については、 ElastiCache (Redis OSS) 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 が T3-Standard キャッシュノードをサポートする ElastiCache ようになりました	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、ベースラインレベルのCPUパフォーマンスを提供し、蓄積されたクレジットを使い果たすまで、いつでもCPU使用状況をバーストできます。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 が、既存の ElastiCache (Redis OSS) サーバーの 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 (Redis OSS) 5.0.6 では、新しいトークンを設定および口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 OSS AUTH コマンドによるユーザーの認証」を参照してください。	2019 年 10 月 30 日
Amazon が Amazon OSS の Redis からのオンラインデータ移行をサポートする ElastiCache ようになりました EC2	オンライン移行を使用して、Amazon OSSのセルフホスト Redis から Amazon EC2 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 (Redis OSS) は、Redis OSS クラスターモードのオンライン垂直スケーリングを導入します。	シャード Redis OSS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。ElastiCache （ Redis OSS) 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する間、ノードタイプを変更してクラスターのサイズを変更します。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 (Redis OSS) では、Amazon ElastiCache (Redis OSS) 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。	この機能を使用すると、すべての読み取りトラフィックを 1 つのクラスターレベルのエンドポイント経由で ElastiCache (Redis OSS) クラスターに誘導して、負荷分散と高可用性を活用できます。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 (Redis OSS) では、ユーザーが独自のスケジュールでサービス更新を適用できるようになりました。	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、特にピークビジネスフロー中のサービスの中断を最小限に抑え、クラスターが で 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もコンプライアンスを維持できます。詳細については、「Amazon のセルフサービス更新 ElastiCache 」と「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 標準リザーブドインスタンスの提供: 一部前払い、すべて前払い、および前払いなし。	リザーブドインスタンスを使用すると、 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 (Redis OSS) Redis OSSクラスターあたり最大 250 ノードのサポート	ノードまたはシャードの制限は、 ElastiCache (Redis OSS) クラスターあたり最大 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 (Redis OSS) は、すべての T2 ノードで自動フェイルオーバーとバックアップ、復元をサポートします。	ElastiCache (Redis OSS) では、すべての T2 ノードで、自動フェイルオーバー、スナップショットの作成、バックアップと復元のサポートが導入されています。詳細については、 ElastiCache (Redis OSS) Backup and Restore and Snapshot を参照してください。	2018 年 11 月 19 日
ElastiCache (Redis OSS) M5 ノードと R5 ノードのサポート	ElastiCache (Redis OSS) は、AWS Nitro System に基づいて M5 ノードと R5 ノード、汎用インスタンスタイプと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の数のサポート	ElastiCache (Redis OSS) は、クラスターのダウンタイムのないクラスターからのリードレプリカの追加と削除のサポートを追加しました。このリリースでのこれらの変更やその他の変更の詳細については、 ElastiCache (Redis OSS) 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。リファレンスIncreaseReplicaCountの DecreaseReplicaCount および も参照してください。 ElastiCache API	2018 年 9 月 17 日

FedRAMP コンプライアンス証明書	ElastiCache (Redis OSS) が FedRAMP コンプライアンスの認定を受けるようになりました。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
Valkey または Redis OSS (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 (Redis OSS) では、Valkey または Redis OSS (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 コンプライアンス証明書	ElastiCache (Redis OSS) がPCIDSSコンプライアンスの認定を受けるようになりました。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 (Redis OSS) 4.0.10 のサポート	ElastiCache (Redis OSS) OSS は Redis 4.0.10 をサポートするようになりました。これには、暗号化とオンラインクラスターのサイズ変更の両方が 1 つのバージョンで含まれています。詳細については、ElastiCache (Redis OSS) 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編され、Redis OSS (ElastiCache (RedisOSS) ユーザーガイド) と Memcached (ElastiCache (Memcached) ユーザーガイド) のユーザーガイドがそれぞれ異なるようになりました。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
E ngineCPUUtilization メトリクスのサポート	ElastiCache (Redis OSS) は、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 を追加しました。詳細については、「Redis のメトリクス OSS」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ElastiCache ユーザーガイドの重要な変更点を示しています。

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 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が追加されました。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ー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。 • 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
	• 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	ElastiCache EU (パリ) 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。 • 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	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。 • 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	このリリースでは、Service Linked Roles () のサポート ElastiCache が追加されましたSLR。詳細については、次を参照してください。 • Amazon のサービスリンクロールの使用 ElastiCache • 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	この ElastiCache のリリースでは、 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。 • サポートされているキャッシュノードの種類 • Memcached ノードタイプ固有のパラメータ • Redis OSSノードタイプ固有のパラメータ	2017 年 11 月 20 日
ElastiCache (Redis OSS) 3.2.10 とオンラインリシャーディングのサポート	Amazon ElastiCache (Redis OSS) が ElastiCache (Redis OSS) 3.2.10 のサポートを追加しました。ElastiCache （ Redis OSS) では、受信 I/O リクエストを処理し続ける間、クラスターにシャードを追加または削除するための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。詳細については、次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 • Valkey または Redis のオンライン再シャーディング OSS (クラスターモードが有効)	2017 年 11 月 9 日
HIPAA 適格性	ElastiCache (Redis OSS) 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。 • Amazon のコンプライアンス検証 ElastiCache • Amazon のデータセキュリティ ElastiCache	2017 年 11 月 2 日

変更	説明	変更日
ElastiCache (Redis OSS) 3.2.6 と暗号化のサポート	ElastiCache ElastiCache (Redis OSS) 3.2.6 のサポートが追加されました。これには、次の 2 つの暗号化機能が含まれています。 • 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。 詳細については、次を参照してください。 • Amazon のデータセキュリティ ElastiCache • サポートされているエンジンとバージョン	2017 年 10 月 25 日
接続パターンのトピック	ElastiCache ドキュメントには、Amazon 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が追加されていますVPC。 詳細については、「 ユーザーガイド」のAmazon で ElastiCache キャッシュにアクセスするためのアクセスパターン VPC「」を参照してください。ElastiCache	2017 年 4 月 24 日
Memcached 1.4.34 のサポート	ElastiCache は、以前の Memcached バージョンに多数の修正を組み込む Memcached バージョン 1.4.34 をサポートしています。 詳細については、「」の「Memcached 1.4.34リリースノート」を参照してくださいGitHub。	2017 年 4 月 10 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	ElastiCache は、レプリケーションをサポートするRedis OSSクラスターでの自動フェイルオーバーをテストするためのサポートを追加します。詳細については、次を参照してください。 • 「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。 • TestFailover ElastiCache APIリファレンス の「」。 • AWS CLI リファレンスの「test-failover」	2017 年 4 月 4 日
拡張 Redis OSS 復元	ElastiCache は、クラスターのサイズ変更による拡張 Redis OSSバックアップと復元を追加します。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と の場合CLI、この機能はシャードの数ではなく、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 OSSスロット設定もサポートされています。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。	2017 年 3 月 15 日
新しい Redis OSSメモリ管理パラメータ	ElastiCache に新しい Redis OSSパラメータ が追加されreserved-memory-percent 、予約済みメモリの管理が容易になります。このパラメータは、 ElastiCache (Redis) のすべてのバージョンで使用できますOSS。詳細については、次を参照してください。 • Valkey と Redis の予約済みメモリの管理 OSS • Redis 3.2.4 OSS の新しいパラメータ	2017 年 3 月 15 日

変更	説明	変更日
Memcached 1.4.33 のサポート	ElastiCache では、Memcached バージョン 1.4.33 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。 • Memcached バージョン 1.4.33 • Memcached 1.4.33 で追加されたパラメータ	2016 年 12 月 20 日
欧州西部 (ロンドン) リージョンのサポート	ElastiCache は、EU (ロンドン) リージョンのサポートを追加します。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。 • 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	ElastiCache は、カナダ (モントリオール) リージョンのサポートを追加します。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。 • 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	ElastiCache 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を追加します。詳細については、次を参照してください。 • 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

変更	説明	変更日
米国東部 2 (オハイオ) リージョンのサポート	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 (us-east-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。 • 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
Redis OSS クラスターのサポート	ElastiCache Redis クラスター (拡張) OSS のサポートが追加されました。Redis OSS クラスターを使用しているお客様は、最大 15 個のシャード (ノードグループ) にデータをパーティション化できます。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 OSS クラスターの自動フェイルオーバー時間は、以前のバージョンのフェイルオーバー時間の約 4 分の 1 です。 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。 詳細については、次を参照してください。 • Memcached と Redis の比較 OSS • ElastiCache コンポーネントと機能 — 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 用語	2016 年 10 月 12 日

変更	説明	変更日
M4 ノードタイプサポート	ElastiCache 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を追加します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 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。 • サポートされているキャッシュノードの種類 • Memcached ノードタイプ固有のパラメータ • Redis OSSノードタイプ固有のパラメータ	2016 年 8 月 3 日
ムンバイリージョンのサポート	ElastiCache 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。 • サポートされているキャッシュノードの種類 • Memcached ノードタイプ固有のパラメータ • Redis OSS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	ElastiCache は、Redis OSS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する機能を追加しますElastiCache。詳細については、次を参照してください。 • 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	ElastiCache は、Redis OSSノードタイプをスケールアップする機能を追加します。詳細については、「スケーリング ElastiCache」を参照してください。	2016 年 3 月 24 日

変更	説明	変更日
エンジンの簡単なアップグレード	ElastiCache は、Redis OSS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、「のバージョン管理 ElastiCache」を参照してください。	2016 年 3 月 22 日
R3 ノードタイプのサポート	ElastiCache では、中国 (北京) リージョンと南米 (サンパウロ) リージョンの 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。	2016 年 3 月 16 日
Lambda 関数 ElastiCache を使用したアクセス	Amazon ElastiCache で にアクセスする Lambda 関数の設定に関するチュートリアルを追加しましたVPC。詳細については、「その他の ElastiCache 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 OSS のサポート 2.8.24	ElastiCache は、Redis OSSバージョン 2.8.24 のサポートを追加し、Redis OSS2.8.23 以降に改善が追加されました。バグ修正および不正なメモリー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。 • ElastiCache (Redis OSS) バージョン 2.8.24 (拡張) • Redis 2.8 OSS 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、r3 ノードタイプを持つ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

変更	説明	変更日
Amazon ElastiCache コンソールの変更。	新しい Redis OSSバージョンでは、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 OSS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す。他のオプションと詳細については、「サポートされているエンジンとバージョン」を参照してください。	2015 年 12 月 15 日
Redis 2.8.23 OSS のサポート。	ElastiCache は、Redis OSSバージョン 2.8.23 のサポートを追加し、Redis OSS2.8.22 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 `close-on-slave-write` 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 (Redis OSS) 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 OSS のサポート。	ElastiCache では、Redis OSSバージョン 2.8.22 のサポートが追加され、バージョン 2.8.21 以降の機能強化と改善 ElastiCache が追加されました。改善には以下のものがあります。 • 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。 • その他の 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 OSSパラメータがすべてのクラスターに適用されるrepl-backlog-size ようになりました。 変更および詳細についての詳細なリストについては、「ElastiCache (Redis OSS) バージョン 2.8.22 (拡張)」を参照してください。 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。	2015 年 9 月 28 日

変更	説明	変更日
Memcached 1.4.28 がサポートされました。	ElastiCache では、Memcached バージョン 1.4.24 のサポートと、バージョン 以降の Memcached の改善が追加されました1.4.14。このリリースでは、バックグラウンドタスクとして最近使用した (LRU) キャッシュ管理、ハッシュアルゴリズムとして jenkins または murmur3 の選択、新しいコマンド、その他のバグ修正のサポートが追加されました。詳細については、「Memcached リリースノート」を参照してください。 .	2015 年 8 月 27 日
5.6 を使用した Memcached Auto Discovery PHP のサポート	Amazon のこのリリースでは、PHPバージョン 5.6 の Memcached Auto Discovery クライアントのサポート ElastiCache が追加されました。詳細については、「の ElastiCache クラスタークライアントのソースコードのコンパイル PHP」を参照してください。	2015 年 7 月 29 日
Redis OSS のサポート 2.8.21	ElastiCache では、Redis OSSバージョン 2.8.21 のサポートと、バージョン 以降の Redis OSSの改善が追加されました2.8.19。この Redis OSSリリースには、いくつかのバグ修正が含まれています。詳細については、「Redis 2.8 OSS リリースノート」を参照してください。	2015 年 7 月 29 日
新しいトピック: 外部 ElastiCache からのアクセス AWS	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、「 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

変更	説明	変更日
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置換、ElastiCache：NodeReplacementScheduled、ElastiCache：NodeReplacementRescheduled、および ElastiCache: に関する 3 つのメッセージを追加しますNodeReplacementCanceled。 ノードの交換が予定されているとき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 OSS v. 2.8.19 のサポート。	ElastiCache では、バージョン 2.8.6 以降の Redis OSSバージョン 2.8.19 と Redis OSSの改善がサポートされています。このサポートによって以下が実現します。 • Redis OSS コマンド 、PFADD、PFCOUNT および を含む HyperLogLog データ構造PFMERGE。 • 新しいコマンド ZRANGEBYLEX、ZLEXCOUNT、および を使用したレキシコグラフィック範囲クエリZREMRANGEBYLEX。 • 多数のバグ修正が導入されました。つまり、バックグラウンドセーブ (bgsave) 子プロセスが予期せず終了SYNCしたときにプライマリノードがプライマリに失敗することで、プライマリノードが古いデータをレプリカノードに送信できないようにしています。 の詳細については HyperLogLog、「Redis OSSの新しいデータ構造： HyperLogLog」を参照してください。 PFADD、、PFCOUNTおよび の詳細についてはPFMERGE、「Redis OSSドキュメント」を参照して、「」をクリックしますHyperLogLog。	2015 年 3 月 11 日
コスト配分タグのサポート	ElastiCache は、コスト配分タグのサポートを追加します。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。	2015 年 2 月 9 日

変更	説明	変更日
AWS GovCloud（米国西部) リージョンのサポート	ElastiCache は AWS GovCloud（米国西部) (us-gov-west-1) リージョンのサポートを追加します。	2015 年 1 月 29 日
欧州 (フランクフルト) リージョンのサポート	ElastiCache は、欧州 (フランクフルト) (eu-central-1) リージョンのサポートを追加します。	2015 年 1 月 19 日
Redis OSS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のマルチ AZ のサポートを Redis レOSSプリケーショングループのリードレプリカに追加します。 はレプリケーショングループのヘルス ElastiCache を監視します。プライマリが失敗した場合、 はレプリカ ElastiCache を自動的にプライマリに昇格し、はレプリカを置き換えます。詳細については、「Valkey と Redis でマルチ AZ ElastiCache を使用して のダウンタイムを最小限に抑える OSS」を参照してください。	2014 年 10 月 24 日
AWS CloudTrail サポートされているAPI通話の口グ記録	ElastiCache は、 を使用してすべての ElastiCache API通話を口グ AWS CloudTrail に記録するためのサポートを追加します。詳細については、「を使用した Amazon ElastiCache API コールの口グ記録 AWS CloudTrail」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 は、追加の汎用 (T2) インスタンスのサポートを追加します。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 ElastiCache パラメータの設定」を参照してください。	2014 年 9 月 11 日
Memcached 用の柔軟なノード配置のサポート	ElastiCache では、複数のアベイラビリティーゾーンにまたがる Memcached ノードの作成のサポートが追加されました。	2014 年 7 月 23 日

変更	説明	変更日
サポートされる新しいインスタンスサイズ	ElastiCache 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を追加します。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 ElastiCache パラメータの設定」を参照してください。	2014 年 7 月 1 日
PHP 自動検出	PHP バージョン 5.5 自動検出のサポートが追加されました。	2014 年 5 月 13 日
Redis OSSクラスターのバックアップと復元	このリリースでは、 ElastiCache により、お客様は Redis OSS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、特定の時点におけるクラスターのコピーであり、クラスターメタデータと Redis OSS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。	2014 年 4 月 24 日
Redis OSS 2.8.6	ElastiCache は Redis 2.6.13 に加えて Redis OSS2.8.6 OSS をサポートしています。Redis OSS2.8.6 を使用すると、部分的な再同期と、常に使用できるユーザー定義の最小数のリードレプリカがサポートされるため、リードレプリカの耐障害性と耐障害性が向上します。Redis 2.8.6 OSS 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。	2014 年 3 月 13 日

変更	説明	変更日
Redis OSS キャッシュエンジン	ElastiCache は、Memcached に加えて Redis OSSキャッシュエンジンソフトウェアを提供します。現在 Redis を使用しているお客様は、Redis ElastiCache OSSスナップショットファイルからの既存のデータを含む新しい Redis OSSキャッシュクラスターを「シードOSS」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。 Redis OSSレプリケーション機能をサポートするために、 は ElastiCache API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 OSSキャッシュノードを使用して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た状態を維持する 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。	2013 年 9 月 3 日
デフォルトの Amazon Virtual Private Cloud (VPC) のサポート	このリリースでは、 ElastiCache は Amazon Virtual Private Cloud () と完全に統合されていますVPC。新規顧客の場合、キャッシュクラスターはVPCデフォルトで Amazon に作成されます。詳細については、「Amazon VPCs と ElastiCache セキュリティ」を参照してください。	2013 年 1 月 8 日
PHP キャッシュノードの自動検出のサポート	キャッシュノード自動検出の初期リリースでは、Java プログラムのサポートが提供されていました。このリリースでは、 ElastiCache がキャッシュノードの自動検出をサポートしていますPHP。	2013 年 1 月 2 日

変更	説明	変更日
Amazon Virtual Private Cloud のサポート (VPC)	このリリースでは、 ElastiCache クラスターを Amazon Virtual Private Cloud () で起動できますVPC。デフォルトでは、新規顧客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。既存の顧客は自分のベースVPCで Amazon に移行できます。詳細については、「Amazon VPCs と ElastiCache セキュリティ」を参照してください。	2012 年 12 月 20 日
キャッシュノード自動検出および新しいキャッシュエンジンバージョン	ElastiCache は、キャッシュノードの自動検出を提供します。これにより、クライアントプログラムはクラスター内のすべてのキャッシュノードを自動的に決定し、これらのノードへの接続を開始および維持できます。 このリリースには、新しいキャッシュエンジンバージョンである Memcached バージョン 1.4.14 が用意されています。この新しいキャッシュエンジンでは、スラブ再分散機能の強化、パフォーマンスとスケーラビリティの大幅な向上、複数のバグ修正が加えられています。設定できる新しいキャッシュパラメータは複数あります。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 ElastiCache パラメータの設定」を参照してください。	2012 年 11 月 28 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。 q	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。