ElastiCache (Memcached) ユーザーガイド

Amazon ElastiCache

API バージョン 2015-02-02

Amazon ElastiCache: ElastiCache (Memcached) ユーザーガイド

Table of Contents

Amazon ElastiCache (Memcached) とは

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

Amazon は 2 つの ElastiCache 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、Redis OSS エンジンと Memcached エンジンの両方で動作します。関心のあるエンジンのガイドを使用してください。必要なエンジンがわからない場合は、このガイドの「Memcached と Redis OSS の独自設計型キャッシュの比較」を参照してください。

サーバーレスキャッシュ

ElastiCache (Memcached) 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 Memcached ベースのキャッシュを簡単に追加して操作できます。 ElastiCache （ Memcached) Serverless を使用すると、可用性の高いキャッシュを 1 分以内に作成できるため、インスタンスのプロビジョニングやノードやクラスターの設定が不要になります。デベロッパーは、 ElastiCache コンソール、SDK、または CLI を使用してキャッシュ名を指定することで、サーバーレスキャッシュを作成できます。

ElastiCache 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。 は、アプリケーションで使用されるキャッシュのメモリとコンピューティングを ElastiCache いつでもモニタリングします。 と は、アプリケーションのニーズに合わせて容量を自動的にスケーリングします。 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 Software. ElastiCache manages ハードウェアプロビジョニングを抽象化することで、 モニタリング、 ノードの置換、 とソフトウェアのパッチ適用を自動的かつ透過的に行います。 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 (Memcached) Serverless は Memcached 1.6 以降と互換性があります。

Memcached クラスター ElastiCache 用に独自の を設計する

ElastiCache (Memcached) 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 Memcached クラスターを で設計することを選択できます ElastiCache。これにより、クラスターのノードタイプ、ノード数、ア AWS ベイラビリティーゾーン間の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 ElastiCache 運用できます。 ElastiCache はフルマネージドサービスであるため、クラスターのハードウェアのプロビジョニング、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 (Memcached) 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性が向上します。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、クロス AZ 可用性で運用するかを選択できます。独自のクラスターを設計するときは、キャッシュにアプリケーションが必要とする十分な容量が確保されるように、ノードの種類と数を正しく選択する必要があります。Memcached クラスターに新しいソフトウェアパッチを適用するタイミングも選択できます。

独自の ElastiCache (Memcached) を設計する場合、Memcached 1.4 以降を実行することを選択できます。

仕組み

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

キャッシュとキャッシュエンジン

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

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

トピック

ElastiCache (Memcached) の仕組み

ElastiCache (Memcached) サーバーレス

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

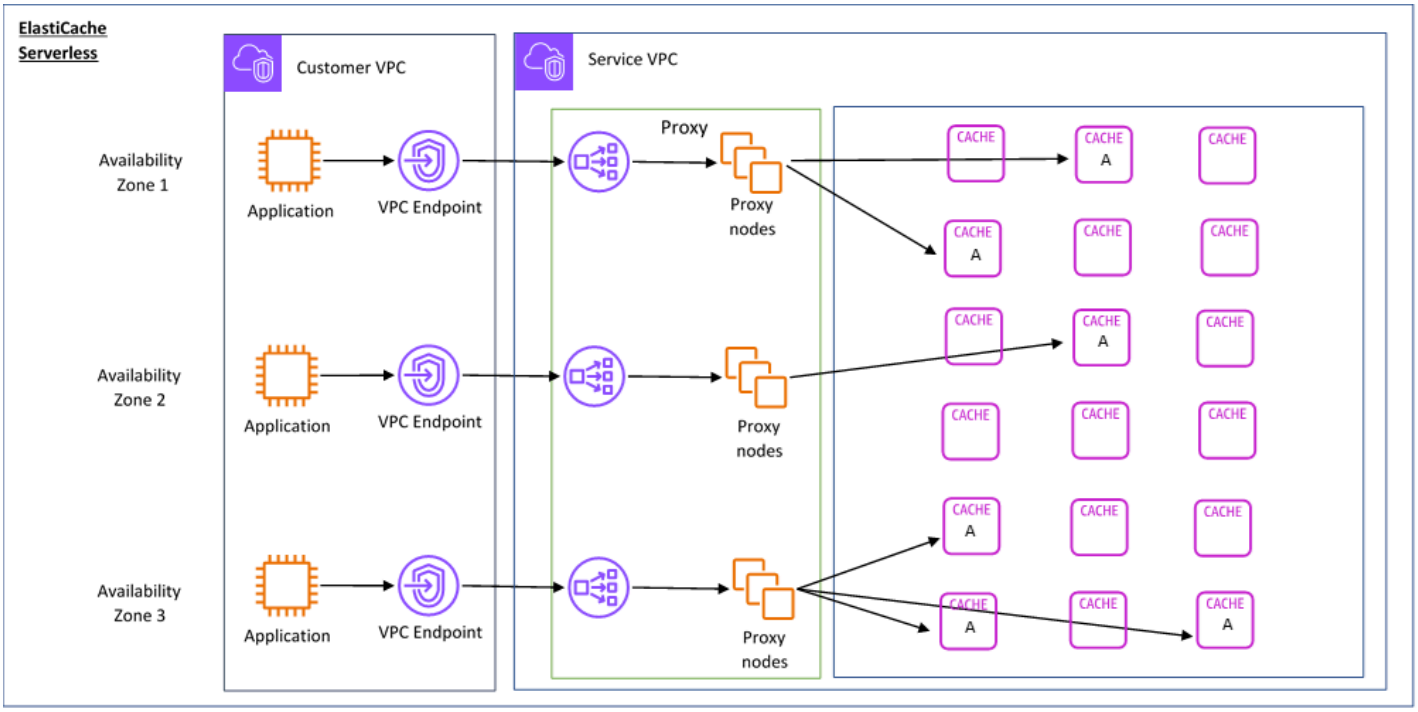
> **Note**
>
> ElastiCache (Memcached) Serverless は、TLS をサポートする Memcached クライアントとのみ互換性があります。

主な利点

- 容量計画なし： ElastiCache Serverless を使用すると、容量を計画する必要がなくなります。ElastiCache Serverless はキャッシュのメモリ、コンピューティング、ネットワーク帯域幅の使用率を継続的にモニタリングし、垂直方向と水平方向の両方にスケーリングします。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および使用されるデータに対して料金が発生します。料金ディメンション を参照してください。
- 高可用性： ElastiCache サーバーレスは、高可用性を実現するために複数のアベイラビリティーゾーン (AZ) 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% の可用性 SLA を提供します。
- ソフトウェアの自動アップグレード： ElastiCache サーバーレスは、アプリケーションの可用性に影響を与えずに、キャッシュを最新のマイナーおよびパッチソフトウェアバージョンに自動的にアップグレードします。新しい Memcached メジャーバージョンが利用可能になると、ElastiCache から通知が送信されます。

- セキュリティ: サーバーレスでは、転送中および保管中のすべてのデータが暗号化されます。保管中のデータを暗号化するには、サービス管理キーを使用するか、独自のカスタマー管理キーを使用できます。

次の図は、 ElastiCache サーバーレスの仕組みを示しています。



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

ElastiCache Serverless では、アプリケーションが接続する単一の DNS エンドポイントを受け取ります。エンドポイントへの新しい接続をリクエストすると、 ElastiCache Serverless はプロキシレイヤーを介した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 ElastiCache 、アプリケーションの可用性に影響を与えたり、接続をリセットしたりすることなく、キャッシュノードの負荷をインテリジェントに

分散し、スケールアウトして新しいキャッシュノードを追加し、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。

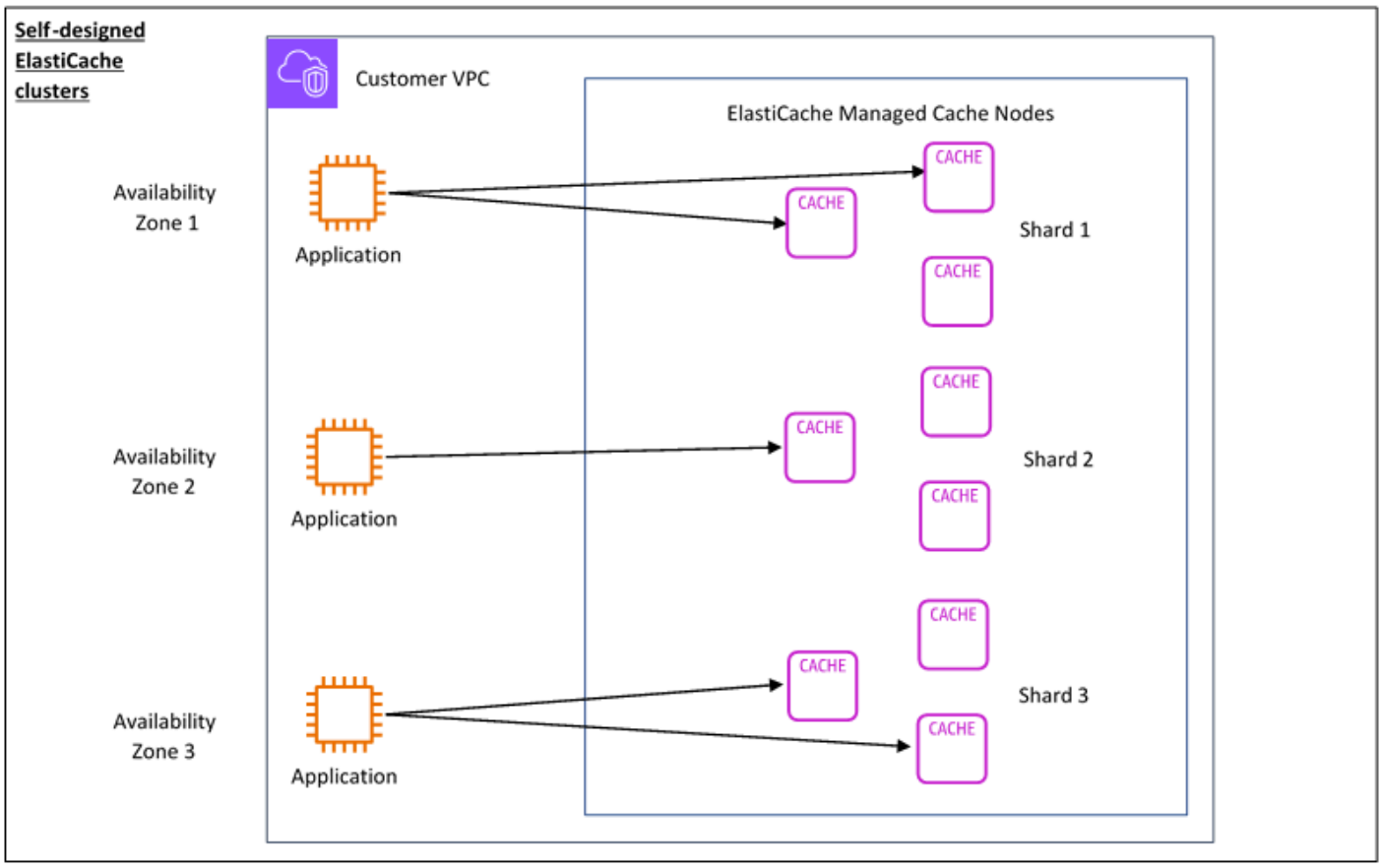
独自設計型 ElastiCache クラスター

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、クラスターの設定とスケーリングをよりきめ細かく制御できます。

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。あるいは、複数の AZ にまたがるノードを含むクラスターを設計して、高可用性を実現することもできます。
- きめ細かな制御: 独自のクラスターを設計すると、キャッシュの設定をより細かく微調整できます。例えば、パラメータグループを使用したエンジンパラメータの設定 を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。ノードを追加して、水平方向にスケールすることもできます。

次の図は、 ElastiCache 独自設計型クラスターの仕組みを示しています。



料金ディメンション

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と ElastiCache 処理単位 (ECPU) で計算されたデータの使用量に対して料金が発生します。独自の ElastiCache (Memcached) 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

ElastiCache Serverless に保存されているデータの料金は、ギガバイト時間 (GB 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継続的にモニタリングし、1 分間に複数回サンプリングし、1 時間あたりの平均を計算して、キャッシュのデータストレージ使用量を GB 時間単位で決定します。各 ElastiCache サーバーレスキャッシュは、最低 1 GB のデータが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 サーバーレスで実行するリクエストに対して、vCPU 時間と転送データの両方を含む単位である ElastiCache 処理単位 (ECPU) でお支払いいただきます。 ECPUs

- 単純な読み取りと書き込みには、転送されるデータの 1 キロバイト (KB) につき 1 ECPU が必要です。例えば、最大 1 KB のデータを転送する GET コマンドは 1 ECPU を消費します。3.2 KB のデータを転送する SET リクエストは 3.2 ECPU を消費します。
- 複数の項目を処理するコマンドは、それに比例して消費する ECPU が増えます。例えば、アプリケーションが 3 つの項目に対してマルチゲットを実行すると、3 ECPU が消費されます。
- 操作する項目が多く、より多くのデータを転送するコマンドは、2 つのディメンションのうち大きい方に基づいて消費する ECPU が決まります。例えば、アプリケーションが 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と、3.2 ECPU を消費します。あるいは、2 KB のデータしか転送しない場合、3 ECPU を消費することになります。

ElastiCache サーバーレスは、ワークロードによって消費`ElastiCacheProcessingUnits`される ECPUs 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

EC2 ノードファミリー、サイズ、ノード数、アベイラビリティーゾーン間の配置を選択することで、独自の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。

一般的な ElastiCache ユースケースと ElastiCache のヘルプ

最新のニュース、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呼び出しでも、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大幅に遅くなります。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。

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、次を参照してください。

- ElastiCache (Memcached) ユーザーガイドの「キャッシュ戦略」

ElastiCache 顧客の声

Airbnb、、PBSEsri 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、その他のElastiCache お客様のユースケースを確認することもできます。

ElastiCache (Memcached) リソース

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

- サービスのハイライトと料金 – 製品詳細ページには、 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 動画セクションには、Amazon ElastiCache for Memcached の紹介、一般的なユースケース、 ElastiCache および (Memcached) を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方法のデモに関する動画があります。
- [はじめに] – 「Amazon ElastiCache (Memcached) の開始方法」セクションには、キャッシュクラスターの作成、キャッシュクラスターへのアクセス認可、キャッシュノードへの接続、キャッシュクラスターの削除のプロセスを実行する例が掲載されています。
- 大規模なパフォーマンス – Amazon ホワイトペーパーの「大規模なパフォーマンス ElastiCache」では、アプリケーションが大規模に正常に機能することを可能にするキャッシュ戦略について説明しています。

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 の AWS CLI 作業、認証情報の提供について説明します AWS 。
- 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 コマンドのすべての AWS CLI について説明します。

アプリケーションプログラムを作成して、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、 for Memcached をサポートする ElastiCache多数の Software Development Kit (SDKs) を提供しています。Java、.NET、PHP、Ruby、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。これらの SDKsは、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。

- ElastiCache API の使用

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

- Amazon ElastiCache API リファレンス

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

実装を管理するためのツール

Amazon EC2 インスタンスに ElastiCache クラスターへのアクセスを許可する ElastiCacheと、ElastiCache クラスターを管理する 、 AWS CLI の AWS Management Console、 AWS SDK for ElastiCache、 ElastiCache および API の 4 つの方法があります。

の使用 AWS Management Console

AWS Management Console は、Amazon ElastiCache (Memcached) を管理する最も簡単な方法です。このコンソールを使用すると、コードを記述しなくても、キャッシュクラスターの作成、キャッシュノードの追加と削除、他の管理タスクを実行することができます。コンソールには、キャッシュエンジンのアクティビティ CloudWatch、メモリと CPU の使用率、およびその他のメトリクスを示す のキャッシュノードパフォーマンスグラフも表示されます。詳細については、この「ユーザーガイド」の特定のトピックを参照してください。

の使用 AWS CLI

に AWS Command Line Interface （ AWS CLI) を使用することもできます ElastiCache。 AWS CLI を使用すると、キャッシュクラスターの起動や停止などのone-at-a-time 操作を簡単に実行できます。また、選択したスクリプティング言語から ElastiCache コマンド AWS CLI に対して を呼び出すこともできます。これにより、繰り返しタスクを自動化できます。の詳細については AWS CLI、「 ユーザーガイド」およびAWS CLI 「 コマンドリファレンス」を参照してください。

AWS SDK の使用

アプリケーション ElastiCache から にアクセスする場合は、 AWS Software Development Kit (SDKs のいずれかを使用できます。SDKs ElastiCache API コールをラップし、 ElastiCache API の低レベルの詳細からアプリケーションを隔離します。開発者が認証情報を指定すれば、SDK ライブラリによって認証とリクエスト署名の処理が自動的に行われます。 AWS SDKs「Amazon Web Services のツール」を参照してください。

ElastiCache API の使用

ElastiCache ウェブサービス API に対してアプリケーションコードを直接記述することもできます。API を使用するときは、HTTP リクエストを構築および認証し、 からの結果を解析し ElastiCache、エラーを処理するために必要なコードを記述する必要があります。API の詳細については、「ElastiCache API の使用」を参照してください。

以下も参照してください。

Amazon ElastiCache (Memcached) デプロイの管理の詳細については、以下を参照してください。

- の使用 ElastiCache
- インターネットトラフィックのプライバシー
- Amazon ElastiCache での口グ記録とモニタリング

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
- 独自のクラスター設計

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ように即座にスケーリングします。 ElastiCache Serverless を使用すると、可用性が高く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、管理する必要がありません。 ElastiCache Serverless は 3 つのアベイラビリティーゾーン間でデータを自動的に冗長的に保存し、99.99% の可用性サービスレベルアグリーメント (SLA) を提供します。ElastiCache は AZs 間で自動データレプリケーションを提供するため、レプリカとカスタムソフトウェアを手動で管理して同期させる必要はありません。

独自設計型クラスター

ElastiCache (Memcached) 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 Memcached クラスターを で設計することを選択できます ElastiCache。これにより、クラスターのノードタイプ、ノード数、ア AWS ベイラビリティーゾーン間の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 ElastiCache 運用できます。 ElastiCache はフルマネージドサービスであるため、クラスターのハードウェアプロビジョニング、モニタリング、ノード交換、ソフトウェアパッチ適用を自動的に管理します。

デプロイオプションの選択

次の場合はサーバーレスキャッシュを選択してください。

- 新規または未知のワークロード用に新しいキャッシュを作成する場合。
- アプリケーションのトラフィックが予測できない場合。
- 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、Memcached を実行しているノードのタイプ、ノードの数、ノードの配置をよりきめ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が大きく変動することを想定しておらず、アプリケーショントラフィックのピークと底を正確に予測できる場合。
- 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。

関連トピック:

- Memcached を実装するための独自の ElastiCache クラスターの設計と管理

Memcached と Redis OSS の独自設計型キャッシュの比較

Amazon は Memcached および Redis OSS キャッシュエンジン ElastiCache をサポートしています。各エンジンにはいくつかのメリットがあります。このトピックの情報を参考にして、要件を満たす最適なエンジンとバージョンを選択してください。

> **Important**
>
> キャッシュ、独自設計型クラスター、またはレプリケーショングループを作成したら、新しいエンジンバージョンにアップグレードできますが、古いエンジンバージョンにダウングレードすることはできません。古いエンジンバージョンを使用する場合は、既存のキャッシュ、独自設計型クラスター、またはレプリケーショングループを削除し、以前のエンジンバージョンで再度作成する必要があります。

見かけ上エンジンは似ています。それぞれのエンジンは、インメモリキー/値ストアです。ただし、実際には大きな違いがあります。

以下がお客様の状況に当てはまる場合は、Memcached を選択します。

- できるだけシンプルなモデルが必要である。

- 複数のコアまたはスレッドを持つ大きなノードを実行する必要がある。
- システムでの需要の増減に応じてノードを追加または削除するスケールアウトおよびスケールイン機能が必要である。
- オブジェクトをキャッシュする必要があります。

次の条件に当てはまる場合は、 ElastiCache (Redis OSS) のバージョンで Redis OSS を選択します。

- ElastiCache (Redis OSS) バージョン 7.0 (拡張)

  Redis OSS Functions 、シャーディング Pub/Sub 、または Redis OSS ACL の改善 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)

  r6gd ノードタイプを使用して、メモリと SSD の間でデータを階層化する機能が必要です。詳細については、データ階層化を参照してください。
- ElastiCache (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)

  Redis OSS ストリーム を使用します。これは、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であり、コンシューマーがブロッキングまたはノンブロッキングの方法でメッセージを使用できるようにするログデータ構造です。

  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)

  暗号化と、Redis OSS (クラスターモードが有効) クラスターへのシャードの動的な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)

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動的に追加または削除する機能をサポートします。

> **Important**
>
> 現在 ElastiCache 、 (Redis 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 OSS のオンラインリシャーディングのベストプラクティス、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
  - Redis OSS 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 (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、「 のスケーリング」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 バージョンの機能に加えて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) 転送時の暗号化」を参照してください。
  - 保管時の暗号化 詳細については、「Amazon ElastiCache (Redis OSS) 保管時の暗号化」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 OSS 2.8.x の機能に加えて以下の機能が必要な場合は、Redis OSS 3.2.4 (クラスターモード) を選択します。

  - 2～500 のノードグループ間でデータを分割する必要がある (クラスターモードのみ)。
  - 地理空間インデックス作成 (クラスターモードまたは非クラスターモード) が必要。
  - 複数のデータベースをサポートする必要がない。
- ElastiCache (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 OSS 2.8.x または Redis OSS 3.2.4 (非クラスターモード) を選択します。

- 文字列、ハッシュ、リスト、セット、ストアドセット、ビットマップなど、複雑なデータ型が必要である。
- インメモリデータセットをソートまたはランク付けする必要がある。
- キーストアの永続性が必要である。
- 読み取り量が多いアプリケーションのために、プライマリからのデータを 1 つ以上のリードレプリカにレプリケートする必要がある。
- プライマリノードが失敗した場合に、自動的なフェイルオーバーが必要である。
- 発行とサブスクライブ (pub/sub) 機能が必要—クライアントにサーバー上のイベントを通知する必要がある。
- 独自設計型クラスターとサーバーレスキャッシュのバックアップおよび復元機能が必要です。
- 複数のデータベースをサポートする必要がある。

Memcached、Redis OSS (クラスターモードが無効）、Redis OSS (クラスターモードが有効) の比較の概要

	Memcached	Redis OSS (クラスターモードが無効）	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 *	3.2.x 以降 - 複雑
		複雑	
データのパーティション化	あり	いいえ	あり
クラスターが変更可能	はい	あり	3.2.10 以降 - 限定
オンラインリシャーディング	なし	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	なし	6.2 以降	6.2 以降
コンプライアンス認定			
コンプライアンス認定			
FedRAMP	はい – 1.6.12 以降	4.0.10 以降	4.0.10 以降
HIPAA	はい – 1.6.12 以降	4.0.10 以降	4.0.10 以降
PCI DSS	あり	4.0.10 以降	4.0.10 以降
マルチスレッド	あり	いいえ	なし
ノードタイプのアップグレード	なし	はい	あり

	Memcached	Redis OSS (クラスターモードが無効)	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	はい	はい	あり
高可用性 (レプリケーション)	なし	はい	あり
自動フェイルオーバー	なし	オプションです。	必須
パブリック/サブ機能	なし	はい	あり
ソートされたセット	なし	はい	あり
バックアップと復元	Serverless Memcached のみ、独自設計型 Memcached クラスターは対象外	はい	あり
地理空間インデックス作成	なし	4.0.10 以降	あり

注意:

文字列、オブジェクト (データベースなど)

*文字列セット、並べ替えられたセット、リスト、ハッシュ、ビットマップ、Hyperloglog

文字列、セット、ソートされたセット、リスト、ハッシュ、ビットマップ、hyperloglog、地理空間インデックス

+ 非推奨、サポート終了、または間もなく終了するバージョンを除外します。

クラスターのエンジンを選択した後は、そのエンジンの最新バージョンを使用することをお勧めします。詳細については、「サポートされている ElastiCache (Memcached) バージョン」または「サポートされている ElastiCache (Redis OSS) バージョン」を参照してください。

Amazon ElastiCache (Memcached) の開始方法

このセクションのトピックでは、 ElastiCache コンソールを使用して Memcached サーバーレスキャッシュを作成、アクセス権の付与、接続、および最後に削除するプロセスについて説明します。

トピック

セットアップ

を設定するには ElastiCache :

トピック

にサインアップする AWS アカウント

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 サービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

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

管理アクセスを持つユーザーを作成する

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

のセキュリティ保護 AWS アカウントのルートユーザー

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

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

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

管理アクセスを持つユーザーを作成する

1. IAM 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権を付与します。

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

管理アクセス権を持つユーザーとしてサインインする

- IAM 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、「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。 AWS サインイン

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権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 （IAMIdentity Center 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、「 ユーザーガイド」の AWS CLI 「 を使用するための の設定 AWS IAM Identity CenterAWS Command Line Interface 」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AWS SDKsツールリファレンスガイド」のIAM「 Identity Center 認証」を参照してください。
IAM	一時的な認証情報を使用して、 AWS CLI、 AWS 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。	「 ユーザーガイド」の「 AWS リソースでの一時的な認証情報の使用IAM」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。 • および ツールについては AWS SDKs、「 AWS SDKs

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、「 IAMユーザーガイド」のIAM「ユーザーのアクセスキーの管理」を参照してください。

関連トピック:

- IAM ユーザーガイド IAMの内容。
- 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を通じて 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) 」の手順に従います。

- IAM ユーザー :
  - ユーザーが担当できるロールを作成します。「 IAMユーザーガイド」のIAM「 ユーザーのロールの作成」の手順に従います。
  - (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「ユーザーへのアクセス許可の追加 (コンソール) 」の指示に従ってください。

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

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

詳細については、次を参照してください。

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

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、「 の開始EC2方法」でEC2インスタンスのセットアップ方法を学習します。
- EC2 インスタンスはキャッシュと同じ にありVPC、セキュリティグループ設定が同じである必要があります。デフォルトでは、Amazon ElastiCache はデフォルトでキャッシュを作成し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、デフォルトのセキュリティグループVPC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 (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の という名前が付けられます。
4. [インバウンド] タブを開き、[編集] をクリックします。

    a. [編集] を選択します。
    b. ルールの追加 を選択します。
    c. タイプ 列で、カスタムTCPルール を選択します。
    d. [ポート範囲] ボックスに、11211 と入力します。
    e. ソースボックスで、ポート範囲 (0.0.0.0/0) を持つ Anywhere 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. [ポート範囲] ボックスに、11212 と入力します。
    i. ソースボックスで、ポート範囲 (0.0.0.0/0) を持つ Anywhere を選択して、Amazon 内で起動する Amazon VPC EC2インスタンスをキャッシュに接続できるようにします。
    j. [保存] を選択します。

ステップ 1: キャッシュを作成する

以下の手順でキャッシュを起動すると、実際に稼働状態になります。サンドボックスで実行されるわけではありません。キャッシュを削除するまで、ElastiCache の標準のキャッシュ使用料が発生します。ここで説明する演習を一気に終わらせ、最後にキャッシュを削除すれば、使用料の総額はごくわずかです (通常 1 ドル未満です)。ElastiCache の使用料の詳細については、「Amazon ElastiCache」を参照してください。

サーバーレスキャッシュの作成

AWS Management Console

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

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. コンソールの左側のナビゲーションペインで、[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「アクティブ」状態になったら、キャッシュへのデータの書き込みと読み取りを開始できます。

AWS CLI を使用して新しいキャッシュを作成するには

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

Linux

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

Windows

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

[Status] 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 でキャッシュの作成が終了したか確認するには、`describe-serverless-caches` コマンドを使用します。

Linux

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

Windows

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

新しいキャッシュを作成したら、「ステップ 2: キャッシュへのデータの読み取りと書き込みを行う」に進んでください。

ステップ 2: キャッシュへのデータの読み取りと書き込みを行う

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

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

設定

開始する前に、アクセス可能な適切なポートがあることを確認してください。

プライマリポート： 11211

読み取り最適化ポート： 11212

サーバーレス Memcached 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、OSS Memcached と同じ整合性保証を持つ書き込みと読み取りを許可します。読み取り最適化ポートを使用すると、書き込みに加えて、結果整合性のある読み込みの低レイテンシーが可能になります。

キャッシュエンドポイントを見つける

AWS Management Console

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

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/elasticache/ で Amazon ElastiCache コンソールを開きます。
2. コンソールの左側のナビゲーションペインで、[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、 describe-serverless-caches コマンドを使用して新しいキャッシュのエンドポイントを検索する方法を示しています。コマンドを実行したら、「Endpoint」フィールドを探します。

Linux

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

Windows

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

OpenSSL を使用して接続する

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

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 転送時の暗号化 (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、以下のように 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 を参照してください

```
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 .NET クライアントを使用して接続する (EnyimMemcachedCore)

https://github.com/cnblogs/EnyimMemcachedCore を参照

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

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

ステップ 3: (オプション) クリーンアップ

AWS Management Console の使用

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. AWS Management Console にサインインして、Amazon ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ElastiCache コンソールのダッシュボードで、削除するキャッシュで実行されているエンジンを選択します。そのエンジンを実行しているすべての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 1 つずつキャッシュを削除できます。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. アクションとして、Delete (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。

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

AWS CLI の使用

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
```

delete-serverless-cache CLI アクションは、サーバーレスキャッシュを 1 つだけ削除します。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 delete-serverless-cache を呼び出します。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、ElastiCacheトピックの「delete-serverless-cache」に関する AWS CLI を参照してください。

これで、「次のステップ」に進むことができます。

次のステップ

ElastiCache については、以下を参照してください。

- の使用 ElastiCache
- スケーリング ElastiCache (Memcached)
- 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: Amazon VPC ElastiCache 内の Amazon にアクセスするための Lambda 関数の設定

このチュートリアルでは、以下の作業を行います。

- us-east-1 リージョンのデフォルトの Amazon Virtual Private Cloud (Amazon VPC) に Amazon ElastiCache キャッシュを作成します。
- ElastiCache キャッシュにアクセスするための Lambda 関数を作成します。Lambda 関数を作成する際には、Lambda 関数で VPC 内のリソースにアクセスできるように、Amazon VPC 内のサブネット ID と VPC セキュリティグループを指定します。このチュートリアルでは、例示のため、この Lambda 関数で UUID の生成、キャッシュへの書き込み、キャッシュからの取得を行います。
- Lambda 関数を手動で呼び出し、VPC 内の ElastiCache キャッシュにアクセスしたことを確認します。

> **Important**
>
> このチュートリアルでは、アカウント内で us-east-1 リージョンのデフォルト Amazon VPC を使用します。Amazon VPC の詳細については、Amazon VPC ユーザーガイドの Amazon VPC の開始方法を参照してください。

トピック

使用を開始する

ステップ 1: ElastiCache キャッシュを作成する

ステップ 1: ElastiCache キャッシュを作成する

このステップでは、 AWS CLI を使用して、アカウントの us-east-1 リージョンのデフォルトの Amazon Virtual Private Cloud に Amazon ElastiCache キャッシュを作成します。 ElastiCache コン

ソールまたは API を使用して ElastiCache サーバーレスキャッシュを作成する方法については、ElastiCache (Memcached) ユーザーガイドクラスターの作成の「」を参照してください。

AWS Management Console

次の AWS CLI コマンドを実行して、us-east-1 リージョンのデフォルト VPC に新しい Memcached クラスターサーバーレスキャッシュを作成します。

Linux

```
aws elasticache create-serverless-cache \
--serverless-cache-name serverlessCacheForLambda \
--region us-east-1 \
--engine memcached
```

Windows

```
aws elasticache create-serverless-cache ^
--serverless-cache-name serverlessCacheForLambda ^
--region us-east-1 ^
--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。がクラスターの作成を完了する ElastiCache までに数分かかる場合があります。

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 serverlessCacheforLambda \
--region us-east-1
```

Windows

```
aws elasticache describe-serverless-caches ^
--serverless-cache-name serverlessCacheforLambda ^
--region us-east-1
```

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。

新しいキャッシュの作成後、「ステップ 2: Lambda 関数を作成する」に進みます。

次のステップ:

ステップ 2: Lambda 関数を作成する

ステップ 2: Lambda 関数を作成する

このステップでは、次の作業を行います。

1. 提供されたサンプルの コードを使用して Lambda 関数のデプロイパッケージを作成します。
2. IAM ロール (実行ロール) を作成します。デプロイパッケージのアップロード時には、このロールを指定する必要があります。これにより、Lambda がロールを引き受け、ユーザーに代わって関数を実行することができます。このロールのアクセス権限ポリシーは、Elastic Network Interface (ENI) をセットアップする権限を AWS Lambda に許可します。これにより Lambda 関数は VPC 内のリソースにアクセスできるようになります。この例では、Lambda 関数は VPC 内の ElastiCache クラスターにアクセスします。
3. デプロイパッケージをアップロードして、Lambda 関数を作成します。

次のステップ

ステップ 2.1: デプロイパッケージを作成する

ステップ 2.1: デプロイパッケージを作成する

現在、Lambda 関数のサンプルコードは Python でのみ提供されています。

Python

次の例の Python コードでは、ElastiCache クラスターに対して項目の読み取り/書き込みを行います。コードを `app.py` という名前のファイルに保存します。コード内の `elasticache_config_endpoint` 値を、ステップ 1 でコピーしたエンドポイントアドレスに必ず置き換えてください。

```
import uuid
```

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

elasticache_config_endpoint = "serverlesscacheforlambda-
ces85m.serverless.use1.cache.amazonaws.com"
target_port = 11211

context = ssl.create_default_context()

memcached_client = Client((elasticache_config_endpoint, target_port),
 tls_context=context)

def lambda_handler(event, context):

    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex

    # put the UUID to the cache
    memcached_client.set("uuid", uuid_in, expire=500, noreply=False)

    # get the item (UUID) from the cache
    result = memcached_client.get("uuid")
    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 Memcached.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Memcached"
```

このコードは Python pymemcache ライブラリを使用してアイテムをキャッシュに格納し、取得します。pymemcache を含むデプロイパッケージを作成するには、次のステップを実行します。

1. app.py ソースコードファイルを含むプロジェクトディレクトリに、pymemacache ライブラリをインストールするフォルダ package を作成します。

   ```
   mkdir package
   ```

2. pip を使用して pymemcache をインストールします。

```
pip install --target ./package pymemcache
```

3. pymemcache ライブラリを含む zip ファイルを作成します。Linux および macOS では、次のコマンドを実行します。Windows では、任意の zip ユーティティを使用して、pymemache ライブラリをルートに置く .zip ファイルを作成します。

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

4. .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 ユーティティを使用して .zip ファイルのルートに app.py を追加します。

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

次のステップ

ステップ 2.2: IAM ロール (実行ロール) を作成する

ステップ 2.2: IAM ロール (実行ロール) を作成する

このステップでは、次の定義済みロールタイプとアクセスポリシーを使用して AWS Identity and Access Management (IAM) ロールを作成します。

- AWS Lambda タイプの AWS サービスロール – このロールでは AWS Lambda ロールを引き受けるアクセス権限が付与されます。
- AWSLambdaVPCAccessExecutionRole – これは、ロールにアタッチするアクセス権限ポリシーです。このポリシーでは、AWS Lambda が ENI を管理するために必要とする、EC2 アクション用のアクセス許可が付与されます。この AWS マネージドポリシーは、IAM コンソールで確認できます。

IAM ユーザーロールの詳細については、『IAM ユーザーガイド』の「ロール (委任とフェデレーション)」を参照してください。

次の手順に従って IAM ロールを作成します。

IAM (実行) ロールを作成するには

1. https://console.aws.amazon.com/iam/ で AWS マネジメントコンソールにサインインして、IAM コンソールを開きます。
2. [Roles] (ロール) を選択してから [Create role] (ロールの作成) を選びます。
   - [信頼されたエンティティのタイプ] で、[AWS のサービス] を選択し、[ユースケース] で、[Lambda] を選択します。これにより、ロールを引き受けるアクセス権限を AWS Lambda サービスに付与します。[Next] (次へ) をクリックします。
   - [アクセス許可を追加] で、**AWSLambdaVPCAccessExecutionRole** を検索し、ポリシーの横にあるチェックボックスをオンにします。
   - [Next] (次へ) をクリックします。
   - [Role Name] (ロール名) では、AWS アカウント内で一意の名前 (lambda-vpc-execution-role など) を使用します。
   - [Create role] (ロールの作成) を選択します。
3. ロールの ARN をコピーします。これは、次のステップで Lambda 関数を作成するときに必要になります。

次のステップ

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

ステップ 2.3: 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 (`AccessMemcached`) を作成します。

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

`role` オプションには、ステップ 2.2 で作成した実行ロールの ARN を使用します。`vpc-config` には、デフォルト VPC のサブネットとデフォルト VPC のセキュリティグループ ID をコンマで区切ったリストを入力します。これらの値は Amazon VPC コンソール にあります。デフォルト VPC のサブネットを検索するには、[VPC] を選択し、次にお使いの AWS アカウント のデフォルト VPC を選択します。この VPC のセキュリティグループを検索するには、[セキュリティ] で [セキュリティグループ] を選択します。us-east-1 リージョンが選択されていることを確認します。

Linux、macOS、Unix の場合:

```
aws lambda create-function \
--function-name AccessMemcached  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role \
--handler app.lambda_handler \
--runtime python3.11  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id \
```

Windows の場合:

```
aws lambda create-function ^
--function-name AccessMemcached  ^
--region us-east-1 ^
--zip-file fileb://path-to/my_deployment_package.zip ^
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role ^
--handler app.lambda_handler ^
--runtime python3.11  ^
--timeout 30 ^
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id ^
```

オプションとして、同じ AWS リージョンの バケットに .zip ファイルをアップロードし、前述のコマンドでそのバケットとオブジェクト名を指定することもできます。以下に示すように、`--zip-file` パラメータを `--code` パラメータで置き換える必要があります。

```
--code S3Bucket=bucket-name,S3Key=zip-file-object-key
```

AWS Lambda コンソールを使用して Lambda 関数を作成することもできます。関数を作成する際には、Lambda 用の VPC を選択し、表示されたフィールドでサブネットとセキュリティグループを選択します。

次のステップ

ステップ 3: Lambda 関数をテストする

ステップ 3: Lambda 関数をテストする

このステップでは、invoke コマンドを使用して Lambda 関数を手動で呼び出します。Lambda 関数を実行すると、UUID を生成し、Lambda コードで指定された ElastiCache クラスターに、その UUID を書き込みます。次に、Lambda 関数はキャッシュから項目を取得します。

1. AWS Lambda invoke コマンドを使用して Lambda 関数 (AccessMemCache) を呼び出す

   Linux、macOS、Unix の場合:

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

   Windows の場合:

   ```
   aws lambda invoke ^
   --function-name AccessMemCache  ^
   --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/AccessMemcached) を選択して、CloudWatch ログの結果を検証します。ログストリームには、以下と同様のコマンドの出力が含まれます。

     ```
     Success: Inserted 05fcf2e4d6c942209acc89ea79b5b15e. Fetched
      05fcf2e4d6c942209acc89ea79b5b15e from Memcached.
     ```

   - AWS Lambda コンソールで結果を確認します。

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 VPC ElastiCache 内の Amazon にアクセスするための Lambda 関数の設定

ElastiCache 動画

以下に、Amazon ElastiCache の基本的な概念と高度な概念を学ぶのに役立つ動画を示します。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
- 上級者向けの動画

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 の最新情報 ElastiCache
- AWS re:Invent 2017: Amazon の新機能 ElastiCache
- DAT204 — AWS NoSQL サービスでのスケーラブルなアプリケーションの構築 (re:Invent 2015)
- DAT207 — Amazon 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 の最新情報 ElastiCache

AWS re:Invent 2019: Amazon の最新情報 ElastiCache

AWS re:Invent 2017: Amazon の新機能 ElastiCache

AWS re:Invent 2017: Amazon の新機能 ElastiCache

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

このセッションでは、NoSQL データベースの利点について説明し、 AWS Amazon DynamoDB と Amazon が提供する主要な NoSQL サービスについて説明します ElastiCache。次に、Expedia と

Mapbox という 2 つの主要なお客様から、そのユースケースとアーキテクチャの課題、設計パターンやベストプラクティスなど、 AWS NoSQL サービスを使用してどのように対処したかについてお聞きします。このセッションを通じて、NoSQL とその強力な機能についての理解を深め、データベース関連の課題に自信を持って対処できるようになります。

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

DAT207 — Amazon 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)

この動画では、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、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 を使用してアプリケーションのレイテンシー ElastiCache を改善し、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することでメリットが得られるさまざまなシナリオとユースケースについて説明し、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加速 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 の ElastiCache ベストプラクティス (re:Invent 2020) で成功するための設計
- Amazon ElastiCache (re:Invent 2019) 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
- ベストプラクティス: Amazon EC2 から Redis OSS クラスターの への移行 ElastiCache (re:Invent 2019)
- Amazon & Amazon ElastiCache Aurora STP11 を使用したファンタジースポーツプラットフォームのスケーリング (re:Invent 2018)
- Amazon を使用したクラウド内の信頼性とスケーラビリティに優れた Redis OSS 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)
- 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 の ElastiCache ベストプラクティス (re:Invent 2020) で成功するための設計

Redis OSS 上に構築されたビジネスクリティカルでリアルタイムのアプリケーションの急激な増加に伴い、可用性、スケーラビリティ、セキュリティが最優先事項となっています。オンラインスケーリング、マルチ AZ 配置全体の高可用性、およびセキュリティ設定で成功 ElastiCache するように Amazon を設定するためのベストプラクティスについて説明します。

Amazon の ElastiCache ベストプラクティス (re:Invent 2020) で成功するための設計

Amazon ElastiCache (re:Invent 2019) 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

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、リアルタイムの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説明します。これにより、 ElastiCache 非常に高いパフォーマンス、高いスケーラビリティ、可用性、セキュリティを実現できます。

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

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

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、Amazon EC2 のセルフホスト Redis OSS からフルマネージド Amazon にデータを簡単に移動できるようになりました。このセッションでは、新しいオンライン移行ツールについて学び、デモを参照し、さらに重要なのは、Amazon へのスムーズな移行のための実践的なベストプラクティスについて学びます ElastiCache。

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

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

Dream11は、インド有数のスポーツテクノロジーのスタートアップ企業です。同社は、ファンタジークリケット、サッカー、バスケットボールなどの複数のスポーツをプレイする 4000 万人以上のユーザーの成長基盤を持っており、現在、50 ミリ秒の応答時間で毎分 300 万リクエストを生成する 100 万人の同時ユーザーにサービスを提供しています。このトークでは、Dream11 CTO の Amit Sharma 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 秒の応答ウィンドウ内に 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。Complete Title: AWS re:Invent 2018: Amazon ElastiCache & Amazon Aurora を使用したファンタジースポーツプラットフォームのスケーリング (STP11)

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

Amazon を使用したクラウド内の信頼性とスケーラビリティに優れた Redis OSS 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 OSS) の機能と機能強化について説明します。Redis OSS 5、スケーラビリティとパフォーマンスの向上、セキュリティとコンプライアンスなど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内の信頼性とスケーラビリティに優れた Redis OSS 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背景の覗き見を提供します ElastiCache。Redis OSS および Memcached 製品を使用した一般的な設計パターンと、お客様がそれらをインメモリデータ処理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方法をご覧ください。 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、裏でご覧ください。Memcached および Redis OSS 製品で一般的な設計パターンと、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ているかをご覧ください。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、裏でご覧ください。Memcached および Redis OSS 製品で一般的な設計パターンと、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ているかをご覧ください。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)

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、裏で詳しく説明します。Memcached および Redis OSS サービスの一般的な設計パターンと、お客様がそれらをインメモリオペレーションにどのように使用し、アプリケーションのレイテンシーとスループットを改善し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)

この動画では、一般的なキャッシュのユースケース、Memcached エンジンと Redis OSS エンジン、ニーズにより適し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 エンジンも比較します。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)。

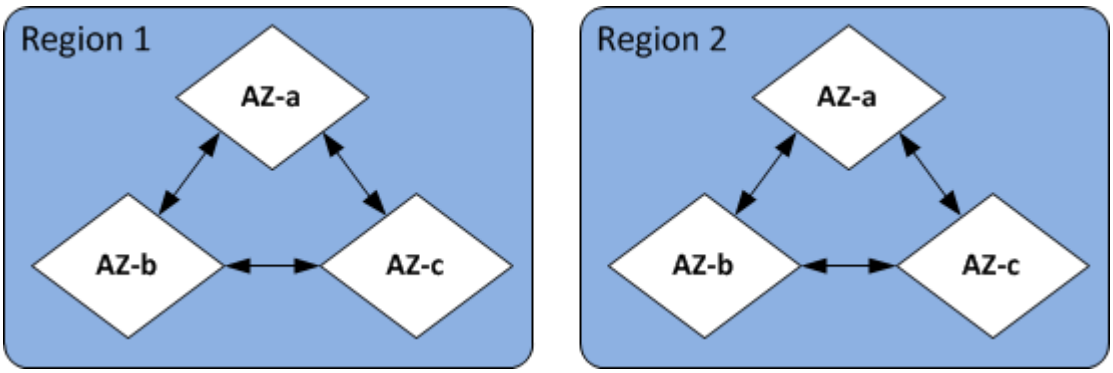
リージョンとアベイラビリティーゾーンの選択

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

AWS リージョンは大きく、地理的に離れた場所に広く分散されています。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。同じ AWS リージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供します。

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

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



リージョンとアベイラビリティーゾーン

トピック

- アベイラビリティーゾーンの考慮事項
- サポートされているリージョンおよびエンドポイント
- ノードの配置
- でのローカルゾーンの使用 ElastiCache

- Outposts の使用

アベイラビリティーゾーンの考慮事項

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。キャッシュを作成するときに、異なるアベイラビリティーゾーンと同じ VPC からサブネットを指定することもElastiCache 、デフォルト VPC からサブネットを自動的に選択することもできます。

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 1 つ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 1 つ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することができます ElastiCache 。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。

1 つ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たい場合は、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 `cross-az` に設定します。
   - 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を加えた数に設定します。
   - NewAvailabilityZones (CLI: `--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。が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。

   - `NumCacheNodes` (CLI: `--num-cache-nodes`) を、この変更を適用した後に必要なアクティブなキャッシュノードの数に設定します。
   - `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。

     一覧表示されたキャッシュノード ID の数は、現在アクティブなノードの数から `NumCacheNodes` の値を引いた数と等しくなる必要があります。
   - (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。

サポートされているリージョンおよびエンドポイント

Amazon ElastiCache は複数の AWS リージョンで利用できます。つまり、要件を満たす場所で ElastiCache クラスターを起動できます。例えば、顧客に最も近い AWS リージョンで を起動したり、特定の法的要件を満たすために特定の AWS リージョンで を起動したりできます。

各リージョンは、他のリージョンと完全に分離されるように設計されています。各リージョンには複数のアベイラビリティーゾーン (AZ) があります。ElastiCache サーバーレスキャッシュは、高可用性を実現するためにus-west-1、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、耐障害性を実現するために、異なる AZs でノードを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「リージョンとアベイラビリティーゾーンの選択」を参照してください。

ElastiCache がサポートされているリージョン

リージョン名/リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ) リージョン us-east-2	elasticache.us-east-2.amazonaws.com	HTTPS	
米国東部 (バージニア北部) リージョン us-east-1	elasticache.us-east-1.amazonaws.com	HTTPS	
US West (N. California) リージョン us-west-1	elasticache.us-west-1.amazonaws.com	HTTPS	
米国西部 (オレゴン) リージョン us-west-2	elasticache.us-west-2.amazonaws.com	HTTPS	

リージョン名/リージョン	エンドポイント	プロトコル	
カナダ (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	
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	

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 (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	
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	

リージョン名/リージョン	エンドポイント	プロトコル	
中東 (アラブ首長国連邦) リージョン me-central-1	elasticache.me-central-1.amazonaws.com	HTTPS	
欧州 (アイルランド) リージョン eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
欧州 (ロンドン) リージョン eu-west-2	elasticache.eu-west-2.amazonaws.com	HTTPS	
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	

リージョン名/リージョン	エンドポイント	プロトコル	
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	
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	

リージョン名/リージョン	エンドポイント	プロトコル	
で AWS GovCloud （ 米国) を使用する方法については ElastiCache、AWS GovCloud 「 (米国) リージョンのサービス： ElastiCache」を参照してください。			

一部のリージョンでは、ノードタイプのサブセットがサポートされています。サポートされているノードタイプの AWS リージョン別の表については、「」を参照してくださいAWS リージョン別のサポート対象ノードタイプ。

リージョン別の AWS 製品とサービスの表については、「リージョン別の製品とサービス」を参照してください。

ノードの配置

Amazon は、クラスターのすべてのノードを 1 つまたは複数のアベイラビリティーゾーン (AZs ElastiCache をサポートしています。さらに、複数の AZs にノードを配置することを選択した場合 (推奨) 、ノードごとに AZ を選択する ElastiCache か、 ElastiCache にノードの選択を許可することができます。

ノードを複数の AZ に配置することによって、1 つの AZ での停電などの障害がシステム全体の障害の原因となる可能性を排除できます。

クラスターの作成時に各ノードの AZ を指定できます。または、既存のクラスターの変更時にノードを追加して AZ を指定することも可能です。詳細については、次を参照してください。

- クラスターの作成
- ElastiCache クラスターの変更
- クラスターへのノードの追加

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、任意の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、VPC 内の他のサブネットと同じように動作します。

Local Zones を使用すると、 ElastiCache クラスターなどのリソースをユーザーに近い複数の場所に配置できます。

ElastiCache クラスターを作成するときに、ローカルゾーンのサブネットを選択できます。Local Zones は、インターネットへの独自の接続を持ち、 AWS Direct Connectをサポートします。したがって、ローカルゾーンで作成したリソースは、非常に低いレイテンシーの通信をローカルユーザーに提供できます。詳細については、「AWS Local Zones」を参照してください。

ローカルゾーンは、 AWS リージョンコードで表され、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。

ローカルゾーン ElastiCache の には、次の制限が適用されます。

- 現時点では、Local Zones では、以下のノードがサポートされています。
  - 現行世代:

    M5 ノードタイプ:
    `cache.m5.large`、`cache.m5.xlarge`、`cache.m5.2xlarge`、`cache.m5.4xlarge`、`cache.m5.`

    R5 ノードタイプ:
    `cache.r5.large`、`cache.r5.xlarge`、`cache.r5.2xlarge`、`cache.r5.4xlarge`、`cache.r5.`

    T3 ノードタイプ: `cache.t3.micro`、`cache.t3.small`、`cache.t3.medium`

ローカルゾーンの有効化

1. Amazon EC2 コンソールでローカルゾーンを有効にします。

   詳細については、Amazon EC2 ユーザーガイドの「Local Zones の有効化」を参照してください。
2. ローカルゾーン内にサブネットを作成します。

   詳細については、Amazon VPC ユーザーガイドの「VPC でサブネットを作成する」を参照してください。
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、「 ユーザーガイド」の「サブネットグループの作成ElastiCache 」を参照してください。
4. ローカルゾーンの 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。

Outposts の使用

AWS Outposts は、 AWS インフラストラクチャ、サービス、APIs、ツールをお客様のオンプレミスに拡張するフルマネージドサービスです。 AWS マネージドインフラストラクチャへのローカルアクセスを提供することで、 AWS Outposts は AWS 、リージョンと同じプログラミングインターフェイ

スを使用してオンプレミスでアプリケーションを構築および実行し、ローカルコンピューティングとストレージリソースを使用してレイテンシーを短縮し、ローカルデータ処理のニーズを低く抑えることができます。Outpost は、お客様のサイトにデプロイされた AWS コンピューティングおよびストレージ容量のプールです。 は、 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 にサブネットを作成し、 ElastiCache クラスターなどの AWS リソースを作成するときに指定できます。

> **Note**
>
> このバージョンでは、次のような制限が適用されます。
>
> - ElastiCache for Outposts は M5 および R5 ノードファミリーのみをサポートします。
> - マルチ AZ (クロス Outpost レプリケーションはサポートされていません)。
> - ElastiCache Outposts の は CoIP をサポートしていません。
> - ElastiCache for Outposts は、次のリージョンではサポートされていません。cn-north-1、cn-northwest-1、ap-northeast-3。

Memcached 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、Memcached キャッシュ を選択します。
3. Memcached キャッシュの作成 を選択します。
4. クラスター設定 で、独自のキャッシュの設計 とクラスターキャッシュ を選択します。クラスターモードは無効 のままにします。次に、キャッシュの名前とオプションの説明を作成します。
5. ロケーション で、オンプレミス を選択します。
6. 「オンプレミス」セクションに、「Outpost ID」というフィールドが表示されます。クラスターを実行する ID を入力します。

   クラスター設定のその他の設定はすべてデフォルトのままにすることができます。
7. 接続 で、新しいサブネットグループの作成 を選択し、VPC ID を入力します。残りはデフォルトのままにして、次へ を選択します。

オンプレミスオプションを設定する

利用可能な Outpost を選択してキャッシュクラスターを選択するか、利用可能な Outposts がない場合は、次の手順を使用して新しい Outpost を作成できます。

[オンプレミスのオプション] の下で:

1. [Memcached の設定] の下で:
   a. [名前]: Memcached クラスターの名前を入力します。
   b. [説明]: Memcached クラスターの説明を入力します。
   c. エンジンバージョンの互換性： エンジンバージョンは Outpost AWS リージョンに基づいています
   d. [ポート]: デフォルトポート (11211) のままにしておきます。理由があって異なるポートを使用する場合は、そのポート番号を入力します。
   e. [パラメータグループ]: ドロップダウンを使用して、デフォルトまたはカスタムパラメータグループを選択します。
   f. [ノードタイプ]: 使用可能なインスタンスは、Outposts 可用性に基づきます。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. [ノード数]: クラスターに必要なノードの数を入力します。
2. 接続 の下：
   a. [サブネットグループ]: リストから [新規作成] を選択します。
      - [名前] - サブネットグループの名前を入力します。
      - [説明] サブネットグループの説明を入力します。
      - [VPC ID]: VPC ID は Outpost VPC と一致する必要があります。
      - [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。使用可能なサブネット ID がない場合は、それらのサブネット ID を作成する必要があります。詳細については、「サブネットの作成」を参照してください。
   b. [作成] を選択します。

Outpost クラスターの詳細の表示

Memcached 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。

AWS CLI での Outposts の使用

AWS Command Line Interface （ AWS CLI) を使用して、コマンドラインから複数の AWS サービスを制御したり、スクリプトを使用して自動化したりできます。 AWS CLI はアドホック (ワンタイム) オペレーションに使用できます。

のダウンロードと設定 AWS CLI

は Windows、macOSで AWS CLI 実行されます。これをダウンロードして設定するには、次の手順に従います。

CLI をダウンロード、インストール、設定するには

1. AWS コマンドラインインターフェイスのウェブページで AWS CLI をダウンロードします。
2. ユーザーガイドの「 AWS CLI のインストール」および「 AWS CLI の設定」の手順に従います。 AWS Command Line Interface

Outposts での AWS CLI の使用

次の CLI オペレーションを使用して、Outposts を使用するキャッシュクラスターを作成します。

- create-cache-cluster – このオペレーションを使用して、 `outpost-mode`パラメータは、キャッシュクラスター内のノードが単一の Outpost に作成されるか、複数の Outposts にまたがって作成されるかを指定する値を受け入れます。

> **Note**
>
> 現時点では、`single-outpost` モードがサポートされています。

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

Memcached を実装するための独自の ElastiCache クラスターの設計と管理

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

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

トピック

- ElastiCache (Memcached) のコンポーネントと機能
- クラスターの管理
- ノードの管理

ElastiCache (Memcached) のコンポーネントと機能

Amazon ElastiCache for Memcached デプロイの主なコンポーネントの概要を以下に示します。

トピック

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に接続された RAM の固定サイズの断片です。各ノードは、Memcached 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 (Memcached)」を参照してください。

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

ノードは、ノードの使用に対してのみ課金される pay-as-you-go ベースで購入できます。または、時間当たりの大幅な割引率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、ほとんどの時間を実行するために多数のリザーブドノードを購入し、 pay-as-you-go ノードの追加が必要になることがある

間はノードを購入できます。リザーブドノードの詳細については、「ElastiCache のリザーブドノード」を参照してください。

Memcached エンジンは自動検出をサポートしています。自動検出は、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です。自動検出を使用すると、手動でアプリケーションを個々のノードに接続する必要はありません。自動的に設定エンドポイントに接続されます。設定エンドポイント DNS エントリには、キャッシュノードエンドポイントごとの CNAME エントリが含まれています。したがって、設定エンドポイントに接続することによって、アプリケーションはクラスター内のすべてのノードに関する情報を入手し、すべてのノードに接続できます。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。詳細については、「自動検出」を参照してください。

ノードの詳細については、「ノードの管理」を参照してください。

ElastiCache (Memcached) クラスター

Memcached クラスター は、単一または複数の ElastiCache 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で分割されます。

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
- クラスターの削除
- クラスターのエレメントの表示
- クラスター間で送受信されるコスト配分タグの追加または削除

詳細については、次の関連トピックを参照してください。

- クラスターの管理 および ノードの管理

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

  これらの制限を超える必要がある場合は、Amazon ElastiCache キャッシュノードリクエストフォーム を使用してリクエストを行います。

- 障害の軽減

  クラスターの耐障害性向上に関する情報。

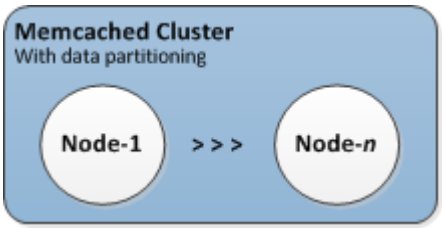
一般的なクラスターの設定

Memcached は、各クラスターに 1～60 個のノードがある AWS リージョンごとに、お客様ごとに最大 300 個の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 個のノードで構成できます。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ノードを配置します。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、「効率的な負荷分散のための ElastiCache クライアントの設定」を参照してください。次の図は、単一ノードと複数ノードの Memcached クラスターの例です。



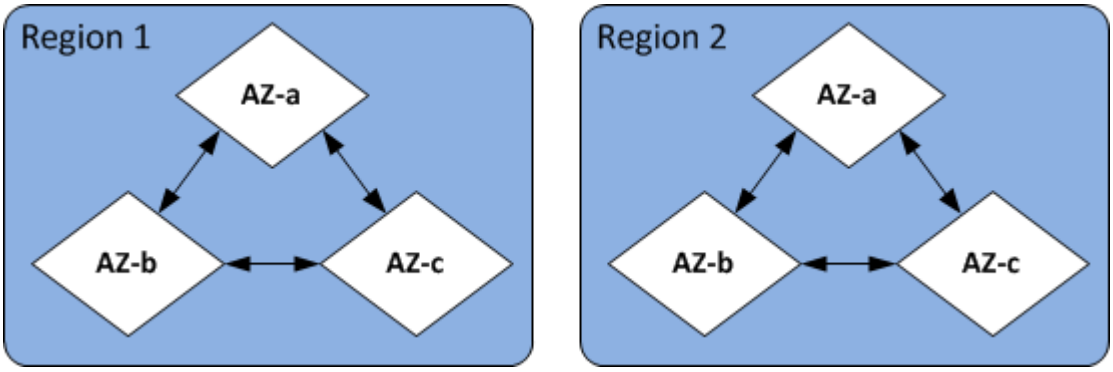
AWS リージョンとアベイラビリティーゾーン

Amazon ElastiCache for Memcached は、世界中の複数の AWS リージョンで利用できます。したがって、ビジネス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 を起動したりできます。

デフォルトでは、 AWS SDKs、 AWS CLI、 ElastiCache API、および ElastiCache コンソールは米国西部 (オレゴン) リージョンを参照します。が新しい AWS リージョンに可用性 ElastiCache を拡張

すると、これらの AWS リージョンの新しいエンドポイントは、HTTP リクエスト、 AWS SDKs、AWS CLIおよび ElastiCache コンソールでも使用できます。

各 AWS リージョンは、他の AWS リージョンから完全に分離されるように設計されています。各リージョン内には複数のアベイラビリティーゾーンがあります。別のアベイラビリティーゾーンでノードを起動して、最大限の耐障害性を実現できます。 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、「」を参照してくださいリージョンとアベイラビリティーゾーンの選択。



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

ElastiCache (Memcached) 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。

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ーのリスト、デフォルト値、変更可能なパラメーターについては、「DescribeEngineDefaultParameters」(describe-engine-default-parameters) を参照してください。

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

ElastiCache 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ホワイトリストに登録されている 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することで、クラスターへのアクセスが許可される Amazon EC2 インスタンスを制御できます。

デフォルトでは、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 インスタンスからのクラスターアクセスを許可できます。Amazon VPC の外部で実行することを選択した場合、セキュリティグループを作成して、特定の Amazon EC2 セキュリティグループ内で実行されている Amazon EC2 インスタンスを承認できます。

ElastiCache サブネットグループ

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

Amazon VPC でクラスターを作成する場合は、キャッシュサブネットグループを指定する必要があります。ElastiCache 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とそのサブネット内の IP アドレスを選択します。

Amazon VPC 環境でのキャッシュサブネットグループの使用方法の詳細については、「Amazon VPC と ElastiCache のセキュリティ」、「アクセスを許可」、および「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 (Memcached) イベント

キャッシュクラスターで重大なイベントが発生すると、 ElastiCache は特定の Amazon SNS トピックに通知を送信します。重要なイベントには、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。

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。

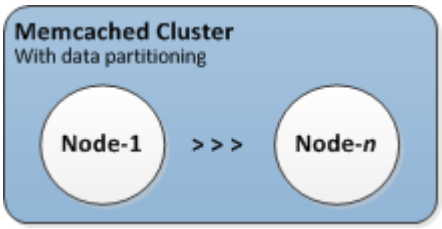
クラスターの管理

クラスターは、1 つ以上のキャッシュノードの集合であり、すべてのノードが Memcached キャッシュエンジンソフトウェアのインスタンスを実行します。クラスターを作成する際に、すべてのノードで使用するエンジンとバージョンを指定します。

次の図は、一般的な Memcached クラスターです。Memcached クラスターには 1 ~ 60 個のノードが含まれており、データを水平にパーティション化できます。

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。

一般的な Memcached クラスターは次のようになります。



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

すべてのクラスターにはクラスター識別子が必要です。クラスター識別子は、お客様が指定するクラスターの名前です。この識別子は、 ElastiCache API および AWS CLI コマンドを操作するとき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するように設計されています。Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) で起動したクラスターには、AWSの外部からアクセスできます。詳細については、「Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン」を参照してください。

サポートされている Memcached バージョンのリストについては、「Supported ElastiCache for Memcached Versions」を参照してください。

ネットワークタイプの選択

ElastiCache は、インターネットプロトコルバージョン 4 および 6 (IPv4 および IPv6) をサポートしているため、以下を受け入れるようにクラスターを設定できます。

- IPv4 接続のみ、
- IPv6 接続のみ、
- IPv4 と IPv6 の両方の接続 (デュアルスタック)

IPv6 は、Nitro システム上に構築されたすべてのインスタンスで 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。IPv6 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。

> **Note**
>
> IPV6/デュアルスタックが使用可能になる前に作成されたクラスターの移行はサポートされていません。新しく作成されたクラスターのネットワークタイプの切り替えもサポートされていません。

ネットワークタイプのサブネットの設定

Amazon VPC でクラスターを作成する場合は、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 アドレスと 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、IPv6-onlyサブネットが IPv6-onlyとして動作するようにする必要があります。

デュアルスタックの使用

キャッシュクラスターを作成し、ネットワークタイプとしてデュアルスタックを選択するときは、IPv4 または IPv6 のいずれかの IP 検出タイプを指定する必要があります。 ElastiCache はデフォルトでネットワークタイプと IP 検出を IPv6 に指定しますが、変更することもできます。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。

既存のすべてのクライアントとの下位互換性を維持するために、IP 検出が導入され、これにより、検出プロトコルでアドバタイズする IP タイプ (IPv4 または IPv6 など) を選択できます。これによ

り、自動検出は 1 つの IP タイプのみに制限されますが、自動検出により、デュアルスタックは、ダウンタイムなしで IPv4 から IPv6 検出 IP タイプへの移行 (またはロールバック) が可能になるため、引き続き有益です。

TLS 対応デュアルスタック ElastiCache クラスター

ElastiCache クラスターで TLS が有効になっている場合、クラスター検出関数 は IP の代わりにconfig get clusterホスト名を返します。 IPs 次に、IPs の代わりにホスト名を使用して ElastiCache クラスターに接続し、TLS ハンドシェイクを実行します。つまり、クライアントは IP 検出パラメータの影響を受けません。TLS が有効なクラスターでは、IP 検出パラメータは優先 IP プロトコルに影響しません。代わりに、使用する IP プロトコルは、DNS ホスト名を解決する際にクライアントがどの IP プロトコルを使用するかによって決まります。

DNS ホスト名を解決する際に IP プロトコルプリファレンスを設定する方法の例については、TLS 対応デュアルスタック ElastiCache クラスター を参照してください。

の使用 AWS Management Console

を使用してキャッシュクラスターを作成する場合 AWS Management Console、接続で、IPv4、IPv6、またはデュアルスタック のいずれかのネットワークタイプを選択します。デュアルスタックを選択した場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

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

CLI の使用

CLI を使用してキャッシュクラスターを作成するときは、 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
```

データ階層化

レプリケーショングループを構成し、r6gd ファミリーのノードタイプを使用するクラスターでは、メモリとローカル SSD (ソリッドステートドライブ) ストレージの間でデータが階層化されます。データ階層化は、データをメモリに保存することに加えて、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、SSD 上のデータにアクセスする際に増加するレイテンシーを許容できるアプリケーションに最適です。

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

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

データ階層化は、アプリケーションワークロードへのパフォーマンスの影響を最小限に抑えるように設計されています。例えば、500 バイトの文字列値を想定した場合に、メモリ内のデータへのリクエストと比較すると、SSD に保存されたデータへのリクエストには平均で 300 マイクロ秒のレイテンシーが生じることが予想されます。

最も大きいデータ階層化ノードサイズ (cache.r6gd.16xlarge) では、単一の 500 ノードクラスターに最大 1 ペタバイトを保存できます (1 つのリードレプリカを使用する場合は 500 TB)。データ階

層化は、 でサポートされているすべての 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。

クラスター内のノードを自動的に識別する

Memcached エンジンを実行するクラスターの場合、ElastiCache では [自動検出] がサポートされます。自動検出は、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です。

> **Note**
>
> 自動検出は、Amazon ElastiCache Memcached で実行されているキャッシュクラスターに追加されます。

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、自動検出は、どちらを選択するかに応じて、IPv4 または IPv6 アドレスのみを返します。詳細については、「ネットワークタイプの選択」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

- 自動検出の利点
- 自動検出の動作
- 自動検出の使用
- キャッシュノードへの手動接続
- クライアントライブラリへの自動検出の追加
- 自動検出機能を持つ ElastiCache クライアント

自動検出の利点

自動検出には、次の利点があります。

- 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。

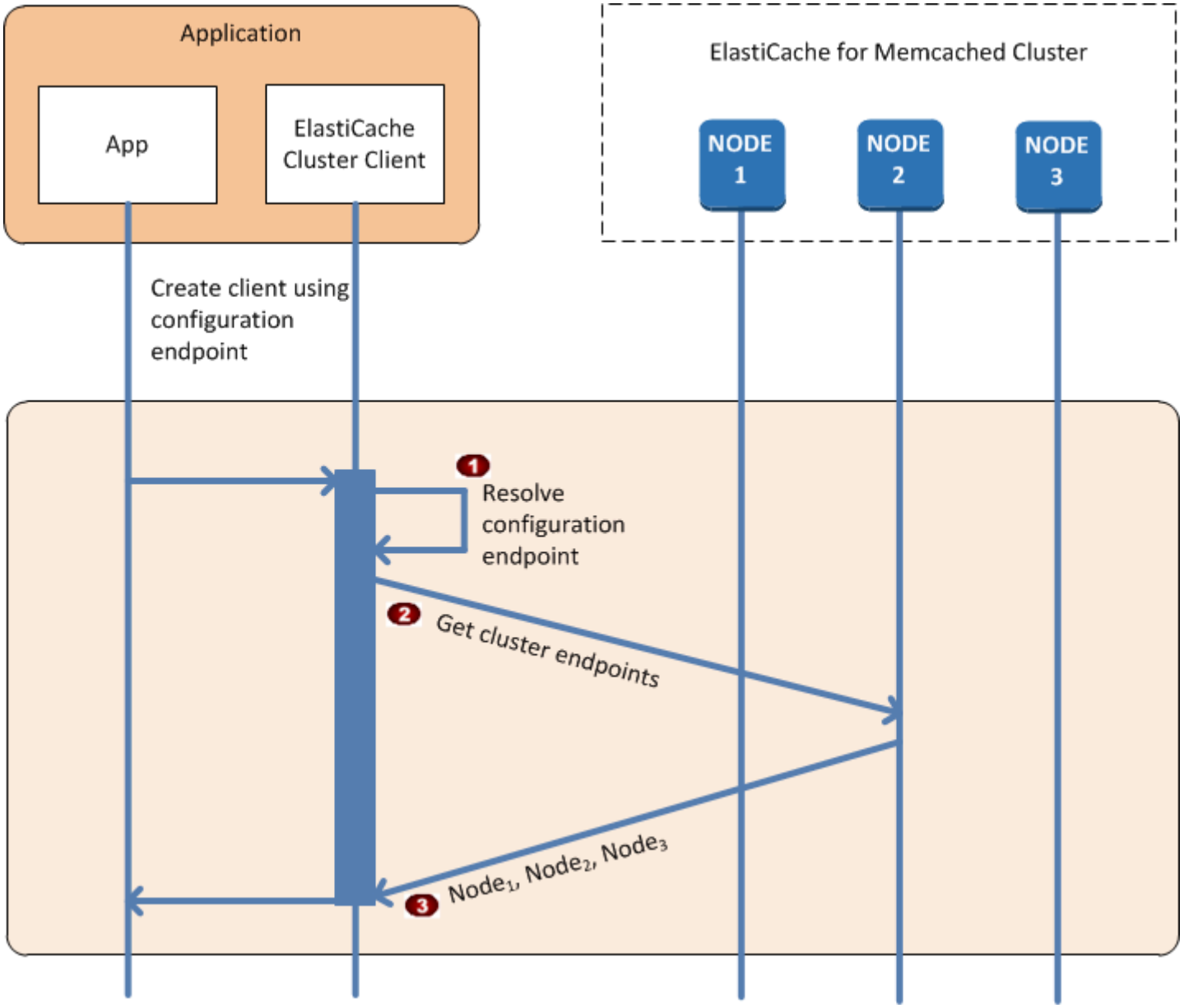
自動検出の動作

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

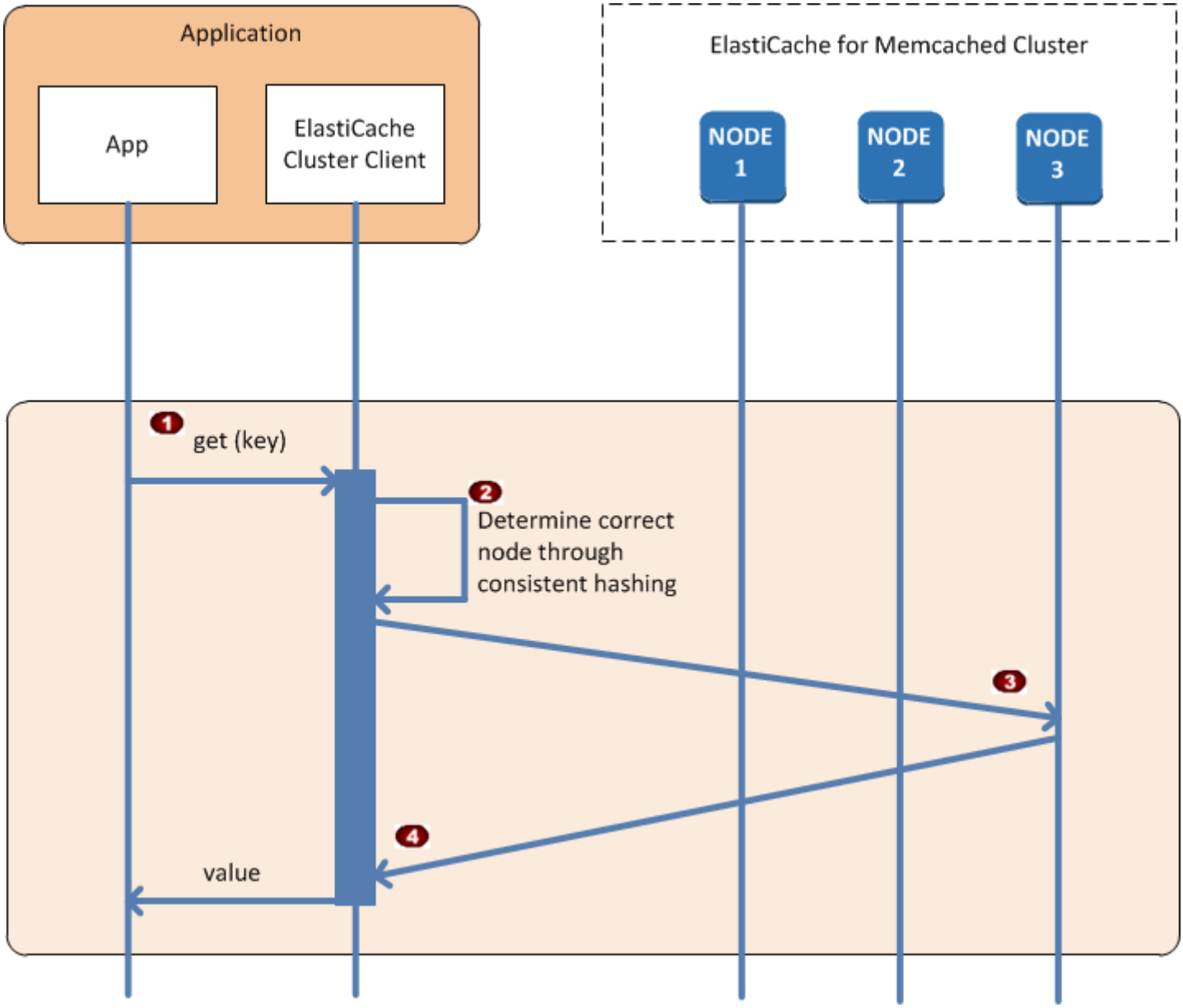
- アプリケーションは、設定エンドポイントの DNS 名を解決します。設定エンドポイントには、すべてのキャッシュノードの CNAME エントリが保持されているため、DNS 名はいずれかのノードに解決されます。その後、クライアントはそのノードに接続できます。
- クライアントは、他のすべてのノードの設定情報をリクエストします。各ノードにはクラスター内のすべてのノードの設定情報が保持されているため、どのノードでも必要に応じて設定情報をクライアントに渡すことができます。

- 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ている場合、ElastiCache クラスタークライアントは、個々のデータ項目を格納する必要があるノードと、それらのデータ項目のクエリを実行する必要があるノードを判断します。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

ノードに障害が発生した場合、ElastiCache がそのノードを停止し、別のノードを起動します。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

自動検出の使用を開始するには、以下のステップに従います。

- ステップ 1: 設定エンドポイントを取得する
- ステップ 2: ElastiCache クラスタークライアントをダウンロードする
- ステップ 3: アプリケーションプログラムを変更する

ステップ 1: 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)」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 AWS CLI for ElastiCache を使用したエンドポイントの検索

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"
        }
    ]
}
```

ステップ 2: ElastiCache クラスタークライアントをダウンロードする

自動検出を利用するには、クライアントプログラムが 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 Cluster Client は、Java、PHP、および .NET 向けが用意されており、すべてのキャッシュノードを検出して接続するのに必要なロジックすべてが含まれています。

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て、Amazon ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ElastiCache コンソールで、[ElastiCache Cluster Client] を選択して [Download] を選択します。

Java 向けの ElastiCache クラスタークライアントのソースコードは、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 ソフトウェアライセンス https://aws.amazon.com/asl の下でリリースされています。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**
>
> PHP 向けの ElastiCache クラスタークライアントを使用するには、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 が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 以上のバイナリをダウンロードします。
> .NET 向けの ElastiCache Cluster Client を使用するには、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 .NET のインストール」を参照してください。

ステップ 3: アプリケーションプログラムを変更する

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、Java、PHP、および .NET 向けの ElastiCache Cluster Client を使用する方法を示します。

> **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」のない CNAME またはエンドポイントは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。

Java 向けの ElastiCache クラスタークライアントの使用

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、キャッシュにデータ項目を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");
    }
}
```

PHP 向けの ElastiCache クラスタークライアントの使用

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、キャッシュにデータ項目を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。

PHP 向けの ElastiCache クラスタークライアントを使用するには、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。

```
<?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 Cluster Client を使用する方法の例については、「Using in transit encryption with PHP and Memcached」を参照してください。

ElastiCache Cluster Client for .NET の使用

> **Note**
>
> ElastiCache .NET Cluster Client は 2022 年 5 月に廃止されました。

ElastiCache の .NET クライアントは、オープンソースとして https://github.com/awslabs/elasticache-cluster-config-net から入手できます。

.NET アプリケーションは、通常、config 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# プログラムは、ElastiCache Cluster Client を使用してクラスター設定エンドポイントに接続し、キャッシュにデータ項目を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。

```
// *****************
// 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
```

キャッシュノードへの手動接続

プログラムクライアントが自動検出を使用していない場合、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` パラメーターを指定して、AWS CLI `aws elasticache describe-cache-clusters` コマンドを使用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。

クライアントライブラリへの自動検出の追加

自動検出の設定情報は、各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、ElastiCache により Memcached ASCII およびバイナリプロトコルに追加され、ElastiCache クラスタークライアントに実装されます。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**
>
> 以下のドキュメントは ASCII プロトコルに関連しています。ただし、`config` コマンドでは ASCII とバイナリの両方がサポートされます。バイナリプロトコルを使用する自動検出サポートを追加する場合、「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`		はい

名前	説明	必須
	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。	
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。	はい

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**
>
> 「AmazonElastiCache:cluster」キーには、クラスターの設定情報が存在するため、変更しないことをお勧めします。このキーを上書きした場合、ElastiCache により設定情報が自動的かつ正しく設定されるまで、クライアントの設定が短時間 (15 秒以内) 不適切にな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
```

各ノードは、CNAME およびプライベート IP アドレスと共に表示されます。CNAME は常に存在します。プライベート IP アドレスを使用できない場合は表示されませんが、その場合もパイプ文字「|」は出力されます。

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。

自動検出機能を持つ ElastiCache クライアント

このセクションでは、ElastiCache PHP と .NET クライアントのインストールと構成について説明します。

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 および .NET Amazon ElastiCache 自動検出クラスタークライアントのインストール、設定、コンパイルを取り上げます。

トピック

- ElastiCache Cluster Client for .NET のインストール
- ElastiCache Cluster Client for PHP のインストール
- PHP 向けの 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

ElastiCache Cluster Client for .NET のインストール

> **Note**
>
> ElastiCache .NET Cluster Client は 2022 年 5 月に廃止されました。

ElastiCache .NET Cluster Client のコードは、オープンソースとして https://github.com/awslabs/elasticache-cluster-config-net から入手できます。

このセクションでは、Amazon EC2 インスタンスで ElastiCache Cluster Client の .NET 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、「自動検出」を参照してください。クライアントを使用するサンプル .NET コードについては、「Auto discovery with DotNET」を参照してください。

トピック

- .NET のインストール
- ElastiCache .NET Cluster Client for ElastiCache のダウンロード
- NuGet を使用した AWS アセンブリのインストール

.NET のインストール

AWS .NET SDK for ElastiCache を使用するには、.NET 3.5 以降がインストールされている必要があります。.NET 3.5 以降がインストールされていない場合は、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。

ElastiCache .NET Cluster Client for ElastiCache のダウンロード

ElastiCache .NET Cluster Client のダウンロード

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ナビゲーションペインで [ElastiCache Cluster Client] をクリックします。
3. [Download ElastiCache Memcached Cluster Client] リストで、[.NET] を選択し、[Download] をクリックします。

NuGet を使用した AWS アセンブリのインストール

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 ではアセンブリの依存関係が認識され、必要なすべてのファイルが自動的にインストールされます。NuGet によってインストールされるアセンブリは、`Program Files` などの一元的な場所ではなく、ソリューションと共に保存されるため、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。

NuGet のインストール

NuGet は MSDN の Installation Gallery からインストールできます。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、NuGet は自動的にインストールされます。

NuGet はソリューションエクスプローラーまたはパッケージマネージャコンソールから使用できます。

ソリューションエクスプローラーからの NuGet の使用

Visual Studio 2010 でソリューションエクスプローラーから NuGet 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 でソリューションエクスプローラーから NuGet 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
```

AWSSDK や AWS.Extensions アセンブリなど、NuGet を通じて利用できる各パッケージ用のページを表示するには、NuGet ウェブサイト (http://www.nuget.org) 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サンプルコマンドラインや、NuGet を通じて利用できる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。

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」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、「PHP 向けの ElastiCache クラスタークライアントの使用」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 Cluster Client の削除

インストールパッケージのダウンロード

適切なバージョンの ElastiCache Cluster Client for PHP を使用するには、Amazon EC2 インスタンスにインストールされている PHP 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 ビットバージョンと 32 ビットバージョンのどちらを実行しているかも確認する必要があります。

Amazon EC2 インスタンスにインストールされている PHP の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
  ```

  PHP のバージョンは、次の例のように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は、以下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、「PHP 向けの 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」を参照してください。

Amazon EC2 AMI 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. AWS Management Console にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. [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. [Description] タブで、[AMI:] フィールドを検索します。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 Cluster Client をダウンロードする準備ができました。

ElastiCache Cluster Client for PHP のダウンロード

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ElastiCache コンソールから [ElastiCache Cluster Client] を選択します。
3. [Download Memcached Cluster Client] リストで、PHP バージョンと AMI アーキテクチャに合った ElastiCache クラスタークライアントを選択し、[Download] ボタンを選択します。

   TLS が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 以上のバイナリをダウンロードします。

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 Cluster Client 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 PHP 7.x ~ 8.x のインストール
- 新規ユーザー向けの PHP 5.x のインストール

新規ユーザー向けの PHP 7.x ~ 8.x のインストール

トピック

- PHP 7.x ~ 8.x を Amazon Linux 2 AMI にインストールするには
- PHP 7.x ~ 8.x を Amazon Linux 201609 AMI にインストールするには
- SUSE Linux 15 AMI に PHP 7.x ~ 8.x をインストールするには
- PHP 7.x ~ 8.x を Ubuntu 22.04 AMI にインストールするには

PHP 7.x ~ 8.x を Amazon Linux 2 AMI にインストールするには

> **Note**
>
> 必要に応じて、*PHP-7.x* を、使用しているバージョンに置き換えます。

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

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

3. amazon-linux-extras を使用して PHP 7.x をインストールする

Amazon Linux 2 では、Extras Library を使用してアプリケーションおよびソフトウェア更新をインスタンスにインストールできます。このようなソフトウェア更新は、トピックと呼ばれます。特定のバージョンのトピックをインストールしたり、最新バージョンを使用するためにバージョン情報を省略したりすることができます。詳細については、「Extras library (Amazon Linux 2)」。

以下に示しているのは、その手順です。

a. まず、[amazon-linux-extras] が既にインストールされているかどうか確認します。

b. インストールされていない場合は、次のコマンドを使用してインストールします。

```
sudo yum install -y amazon-linux-extras
```

c. PHP 7.x トピックが Amazon Linux 2 マシンで利用可能であることを確認します。

```
sudo amazon-linux-extras | grep php
```

d. 出力から、すべての PHP 7 トピックを確認し、必要なバージョンを選択します。

```
sudo amazon-linux-extras enable php7.x
```

e. リポジトリから PHP パッケージをインストールします。例:

```
sudo yum clean metadata

sudo yum install php php-devel
```

4. Amazon ElastiCache Cluster Client をダウンロードします。

- ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。

  ElastiCache ダッシュボードの下で、[ElastiCache クラスター] に進んでから、必要な PHP7 バージョンを選択します。

- コマンドラインから、`PHP-7.X` を目的の PHP バージョンに置き換え、`ARCH` を目的のアーキテクチャ (X86 または arm) に置き換えます。PHP 7.4 以上の場合、OpenSSL を目的の OpenSSL バージョン (openssl1.1 または openssl3) に置き換えます。7.4 より上の PHP を使用している場合は、OpenSSL サフィックスを削除します。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.X/
latest-64bit-<ARCH>-<OpenSSL>
```

5. `tar -zxvf` を使用して、ダウンロードしたファイルを抽出します。

```
tar -zxvf latest-64bit-<ARCH>-<OpenSSL>
```

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/modules にコピーします。

```
sudo mv amazon-elasticache-cluster-client.so /usr/lib64/php/modules/
```

7. extension=amazon-elasticache-cluster-client.so をファイル /etc/php.ini に追加する
8. PHP 7.4 以上で ElastiCache Cluster Client をダウンロードした場合は、OpenSSL 1.1.x 以上をインストールします。OpenSSL 1.1.1 の場合のインストール手順:

```
sudo yum -y update
sudo yum install -y make gcc perl-core pcre-devel wget zlib-devel
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

PHP 7.x ~ 8.x を Amazon Linux 201609 AMI にインストールするには

> **Note**
>
> 必要に応じて、*php7.x* を、使用しているバージョンに置き換えます。

1. AMI から新しいインスタンスを起動します。これを行う方法の詳細については、Amazon EC2 ユーザーガイドの「ステップ 1: インスタンスを起動する」を参照してください。
2. 次のコマンドを実行します。

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

3. PHP をインストールする

```
sudo yum install php7.x
```

4. Amazon ElastiCache Cluster Client をダウンロードします。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
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.x/modules/ にコピーします。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.x/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.x.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 15 AMI に PHP 7.x ~ 8.x をインストールするには

> **Note**
>
> 必要に応じて、*php7.x* を、使用しているバージョンに置き換えます。

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

```
sudo zypper refresh
sudo zypper update -y
```

```
sudo zypper install gcc
```

3. PHP をインストールする

```
sudo yum install php7.x
```

または

```
sudo zypper addrepo //download.opensuse.org/repositories/devel:/languages:/php/
openSUSE_Leap_15.3/ php
```

4. Amazon ElastiCache Cluster Client をダウンロードし、<ARCH> を目的のアーキテクチャ (X86 または arm) に置き換えます。SUSE 15 には OpenSSL1.1 が組み込まれているので、PHP 7.4 以上の場合は、OpenSSL1 のクライアントバイナリを選択します。7.4 未満の PHP を使用している場合は、OpenSSL サフィックスを削除します。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl1.1
```

5. latest-64bit を展開します。

```
tar -zxvf latest-64bit-<ARCH>-openssl1.1
```

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
```

PHP 7.x ~ 8.x を Ubuntu 22.04 AMI にインストールするには

> **Note**
>
> 必要に応じて、*php7.x* を、使用しているバージョンに置き換えます。

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

```
sudo apt-get update
sudo apt-get install gcc g++ make zlib1g zlib1g-dev
```

3. PHP をインストールする

   a. PHP 8.1 の場合のインストール手順:

   ```
   sudo apt install php8.1-cli php8.1-dev
   ```

   b. PHP 7.4 の場合のインストール手順:

   ```
   sudo apt -y install software-properties-common
   sudo add-apt-repository ppa:ondrej/php
   sudo apt-get update
   sudo apt -y install php7.4
   ```

4. Amazon ElastiCache Cluster Client をダウンロードし、<ARCH> を目的のアーキテクチャ (X86 または arm) に置き換えます。Ubuntu 22.04 には OpenSSL3 が組み込まれているので、PHP 7.4 以上の場合は、OpenSSL3 のクライアントバイナリを選択します。7.4 未満の PHP を使用している場合は、OpenSSL サフィックスを削除します。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl3
```

5. 最新の 64 ビットを抽出します。

```
tar -zxvf latest-64bit-<ARCH>-openssl3
```

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 PHP 拡張機能ディレクトリ `/usr/lib/php/20190902` にコピーし

ます。この拡張機能ディレクトリが存在しない場合、`php -i | grep extension_dir` を実行して見つけることができます。

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

新規ユーザー向けの PHP 5.x のインストール

トピック

- Amazon Linux AMI 2014.03 (64 ビットおよび 32 ビット) に PHP 5 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 PHP 5 をインストールするには
- Ubuntu Server 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP 5 をインストールするには
- SUSE Linux Enterprise Server 11 AMI (64 ビットまたは 32 ビット) に PHP 5 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 AMI 2014.03 (64 ビットおよび 32 ビット) に PHP 5 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 の依存関係をインストールします。

```
$ sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2 インスタンスと PHP のバージョンに合った適切な 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。URI にはインストールパッケージのダウンロードパスを指定します。

```
$ sudo pecl install <package download path>
```

PHP 5.4、64 ビット Linux 用のサンプルインストールコマンドを次に示します。このサンプルでは、*X.Y.Z* を実際のバージョン番号に置き換えてください。

```
$ 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、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
```

Red Hat Enterprise Linux 7.0 AMI (64 ビットおよび 32 ビット) に PHP 5 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 の依存関係をインストールします。

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

3. Amazon EC2 インスタンスと PHP のバージョンに合った適切な 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。URI にはインストールパッケージのダウンロードパスを指定します。

```
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 Server 14.04 LTS AMI (64 ビットおよび 32 ビット) に PHP 5 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 の依存関係をインストールします。

```
sudo apt-get update
sudo apt-get install gcc g++ php5 php-pear
```

3. Amazon EC2 インスタンスと PHP のバージョンに合った適切な 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。URI にはインストールパッケージのダウンロードパスを指定します。

```
$ 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、PHP クライアントアーティファクト amazon-elasticache-cluster-client.so を、ダウンロードした *.tgz ファイルから手動で抽出し、/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、memcached.ini という名前の新しいファイルを /etc/php5/cli/conf.d ディレクトリに追加し、このファイルに「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
```

SUSE Linux Enterprise Server 11 AMI (64 ビットまたは 32 ビット) に PHP 5 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 の依存関係をインストールします。

```
$ sudo zypper install gcc php53-devel
```

3. Amazon EC2 インスタンスと PHP のバージョンに合った適切な 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。URI にはインストールパッケージのダウンロードパスを指定します。

```
$ 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。また、使用中の PHP がサポートされているバージョンであることも検証します。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

特に CentOS7 や Red Hat Enterprise Linux (RHEL) 7.1 など、一部のシステムでは、libsasl2.so.3 が libsasl2.so.2 に置き換えられました。これらのシステムは、ElastiCache クラスタークライアントをロードする際、libsasl2.so.2 をロードしようとしますが見つけることができません。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 Cluster Client の削除

トピック

- PHP 7 以上の以前のバージョンの削除
- PHP 5 の以前のバージョンの削除

PHP 7 以上の以前のバージョンの削除

PHP 7 以上の以前のバージョンを削除するには

1. インストール手順で先ほど示した該当する 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
```

PHP 5 の以前のバージョンの削除

PHP 5 の以前のバージョンを削除するには

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。

PHP 向けの 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

このセクションでは、ElastiCache Cluster Client for PHP のソースコードを取得しコンパイルする方法について説明します。

GitHub から取得し、コンパイルする必要がある 2 つのパッケージがあります。aws-elasticache-cluster-client-libmemcached および aws-elasticache-cluster-client-memcached-for-php。

トピック

- libmemcached ライブラリのコンパイル
- PHP 用 ElastiCache Memcached Auto Discovery クライアントのコンパイル

libmemcached ライブラリのコンパイル

前提条件ライブラリ

- OpenSSL 1.1.0 以上 (./configure --disable-tls によって TLS サポートが無効になっている場合を除く)。
- SASL (libsasl2、`./configure --disable-sasl` によって SASL サポートが無効になっている場合を除く)。

aws-elasticache-cluster-client-libmemcached ライブラリをコンパイルするには

1. Amazon EC2 インスタンスの起動
2. ライブラリの依存関係をインストールします。

    - Amazon Linux 201509 AMI / Amazon Linux 2 AMI の場合

```
sudo yum -y update
sudo yum install gcc gcc-c++ autoconf libevent-devel make perl-core pcre-devel
 wget zlib-devel
// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

- Ubuntu 14.04 AMI の場合 (OpenSSL 1.1 以上が付属する Ubuntu バージョンには必要ありません)

```
sudo apt-get update

sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev

// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib/libssl.so.1.1 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
```

3. リポジトリをプルし、コードをコンパイルします。

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-
libmemcached.git
cd aws-elasticache-cluster-client-libmemcached
touch configure.ac aclocal.m4 configure Makefile.am Makefile.in
mkdir BUILD
cd BUILD
../configure --prefix=<libmemcached-install-directory> --with-pic --disable-sasl
```

`../configure` の実行で `libssl` (OpenSSL ライブラリ) を見つけられない場合、`PKG_CONFIG_PATH` 環境変数の微調整が必要になることがあります。

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --prefix=<libmemcached-
install-directory> --with-pic --disable-sasl
```

また、TLS を使用していない場合は、以下を実行して無効にできます。

```
make
sudo make install
../configure –prefix=<libmemcached-install-directory> --with-pic --disable-sasl --
disable-tls
```

PHP 用 ElastiCache Memcached Auto Discovery クライアントのコンパイル

以下のセクションでは、ElastiCache Memcached Auto Discovery クライアントをコンパイルする方法について説明します。

トピック

PHP 7 以上用 ElastiCache Memcached クライアントのコンパイル

PHP-7.x を、使用しているバージョンに置き換えます。

PHP をインストールします。

```
sudo yum install -y amazon-linux-extras
sudo amazon-linux-extras enable php7.x
```

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
phpize
mkdir BUILD
CD BUILD
../configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-
memcached-sasl
```

../configure の実行で libssl (OpenSSL ライブラリ) を見つけられない場合、PKG_CONFIG_PATH 環境変数を OpenSSL の .PC ファイルディレクトリへと微調整することが必要になる場合があります。

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --with-libmemcached-dir=<path
 to libmemcached build directory> --disable-memcached-sasl
```

また、TLS を使用していない場合は、以下を実行して無効にできます。

```
make
```

```
make install
../configure --with-libmemcached-dir=<path to libmemcached build directory> --disable-
memcached-sasl --disable-memcached-tls
```

> **Note**
>
> PHP バイナリに libmemcached ライブラリを静的にリンクして、さまざまな Linux プラットフォーム間でできるようにします。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
> ```

PHP 5 用 ElastiCache Memcached クライアントのコンパイル

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/tree/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、Memcached とプロトコルに準拠しています。既存の Memcached 環境で現在使用しているコード、アプリケーション、および広く使用されているほとんどのツールは、このサービスでもシームレスに動作します。

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

- ノードのエンドポイントおよびポート番号を検索する
- 自動検出を使用するための接続
- DNS 名と基になっている IP

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 マネジメントコンソールにサインインし、クラスターで実行し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。

キャッシュノードのエンドポイントおよびポート番号を検索する (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」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。

DNS 名と基になっている 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。ElastiCache を使用すると、DescribeCacheClusters API (また

は、describe-cache-clusters コマンドラインユーティリティ) により、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が返されます。

> **Important**
>
> キャッシュノードエンドポイントに接続するときは、クライアントアプリケーションがキャッシュノードの DNS 名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

キャッシュノードが障害から復帰した場合に、キャッシュノードの DNS 名と IP アドレスの両方が同じままかどうかを ElastiCache が確認します。

非 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元されても、キャッシュノードの DNS 名が変わらないようにします。ただし、基になっているキャッシュノードの 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。ElastiCache を使用するときは、永続的なキャッシュノード接続の使用をお勧めします。クライアント側の DNS キャッシュが複数の場所 (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) で行われる場合があります。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。

クラスターを準備する

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

AWS CloudFormation を使用してElastiCache クラスターを作成することもできます。詳細については、「AWS クラウド形成ユーザーガイド」の「AWS::ElastiCache::CacheCluster」を参照してください。これには、そのアプローチの実装方法に関するガイダンスが含まれています。

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

トピック

- 要件の特定

- ノードサイズの選択

要件の特定

準備

以下の質問に対する回答を確認することで、クラスターの作成を円滑化できます。

ElastiCache サーバーレスサービスまたはインスタンスベースのサービスのどちらを使用しますか？

サーバーレスキャッシュを使用する場合は、VPC、サブネット、セキュリティグループが正しく設定されていることを確認してください。詳細については、「Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン」を参照してください。インスタンスベースの を使用する場合は ElastiCache、引き続きお読みください。

- 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、「Memcached ノードサイズの選択」を参照してください。
- Amazon VPC に基づいて Virtual Private Cloud (VPC) でクラスターを起動しますか?

> **Important**
>
> VPC でクラスターを起動する場合、クラスターの作成を開始する前に、同じ VPC にサブネットグループを作成する必要があります。詳細については、「サブネットおよびサブネットグループ」を参照してください。
> ElastiCache は、Amazon EC2 AWS を使用して内部からアクセスするように設計されています。ただし、Amazon VPC に基づく VPC で起動し、クラスターが VPC にある場合、AWSの外部アクセスを提供できます。詳細については、「Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン」を参照してください。

- パラメーター値をカスタマイズする必要がありますか。

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

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

詳細については、「障害の軽減」を参照してください。

トピック

- メモリとプロセッサの要件
- Memcached クラスターの構成
- スケーリングの要件
- アクセスの要件
- リージョン、アベイラビリティーゾーン、およびローカルゾーンの要件

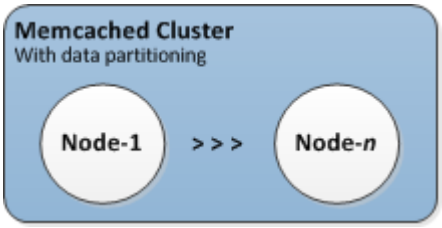
メモリとプロセッサの要件

Amazon の基本的な構成要素はノード ElastiCache 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 個の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 クラスター内のノードを自動的に識別する に任せることができます。



詳細については、「クラスターの管理」を参照してください。

スケーリングの要件

すべてのクラスターは、新しい大きなノードタイプの新規クラスターを作成することでスケールアップすることができます。Memcached クラスターをスケールアップするとき。新しいクラスターは空から起動します。

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 (Memcached)」を参照してください。

アクセスの要件

設計上、Amazon ElastiCache クラスターには Amazon EC2 インスタンスからアクセスします。ElastiCache クラスターへのネットワークアクセスは、クラスターを作成したアカウントに制限されます。したがって、Amazon EC2 インスタンスからクラスターに接続するには、Amazon EC2 インスタンスにクラスターへのアクセスを許可する必要があります。これを行う手順は、EC2-VPC で起動したか、または EC2-Classic で起動したかによって異なります。

クラスターを EC2-VPC で起動した場合、クラスターにネットワーク進入を許可する必要があります。EC2-Classic でクラスターを起動した場合は、インスタンスに関連付けられた Amazon Elastic Compute Cloud セキュリティグループに ElastiCache セキュリティグループへのアクセス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「クラスターへのアクセス」を参照してください。

リージョン、アベイラビリティーゾーン、およびローカルゾーンの要件

Amazon ElastiCache はすべての AWS リージョン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、レイテンシーを短縮できます。クラスターに複数のノードがある場合、複数の異なるアベイラビリティーゾーンに、または Local Zones にノードを配置することで、クラスター上の障害の影響を低減できます。

詳細については、次を参照してください。

- リージョンとアベイラビリティーゾーン

- ローカルゾーン
- 障害の軽減

ノードサイズの選択

クラスターに選択するノードのサイズによって、コスト、パフォーマンス、耐障害性が変わります。

Memcached ノードサイズの選択

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、システムのオーバーヘッドを差し引いた後に、クラスター内のキャッシュノードの数に各ノードの RAM 容量を乗算して計算されます。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 USD	6	41.1	12	635 USD
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 USD	3	42.12	6	442 USD

* 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 の 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 自動検出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がクラスター内のすべてのノードを自動的に識別できるようになります。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」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 に公開される ElastiCache メトリクスを使用して、メモリ使用率、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 メトリクスの使用状況のモニタリング。本稼働のより大きな優れたワークロードの場合は、R5 ノードが最高のパフォーマンスと RAM のコストバリューを提供します。

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 CPU の制約を受けているが、十分なヒット率がある場合は、処理能力のより高いノードタイプで新しいクラスターを設定します。

クラスターの作成

次の例は、、 AWS Management Console、 AWS CLI および ElastiCache API を使用してクラスターを作成する方法を示しています。

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。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」を参照してください。

Memcached クラスターを作成するには、「ステップ 1: キャッシュを作成する」の手順に従います。

クラスターのステータスが [available] になり次第、Amazon EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、「クラスターへのアクセス」および「キャッシュノードへの手動接続」を参照してください。

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

クラスターの作成 (AWS CLI)

を使用してクラスターを作成するには AWS CLI、 `create-cache-cluster` コマンドを使用します。

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

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
```

クラスターの作成 (ElastiCache API)

ElastiCache API を使用してクラスターを作成するには、 `CreateCacheCluster`アクションを使用します。

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

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 (ElastiCache API) を持つ Memcached クラスターを作成します。

改行は読みやすくするために追加しています。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-cluster
    &CacheNodeType=cache.r4.large
    &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>
```

クラスターの詳細を表示する

ElastiCache コンソール、、または ElastiCache API 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。

クラスターの詳細を表示する (コンソール)

Memcached クラスターの詳細は、 ElastiCache コンソール、 AWS CLI の ElastiCache、または ElastiCache API を使用して表示できます。

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詳しく説明します。

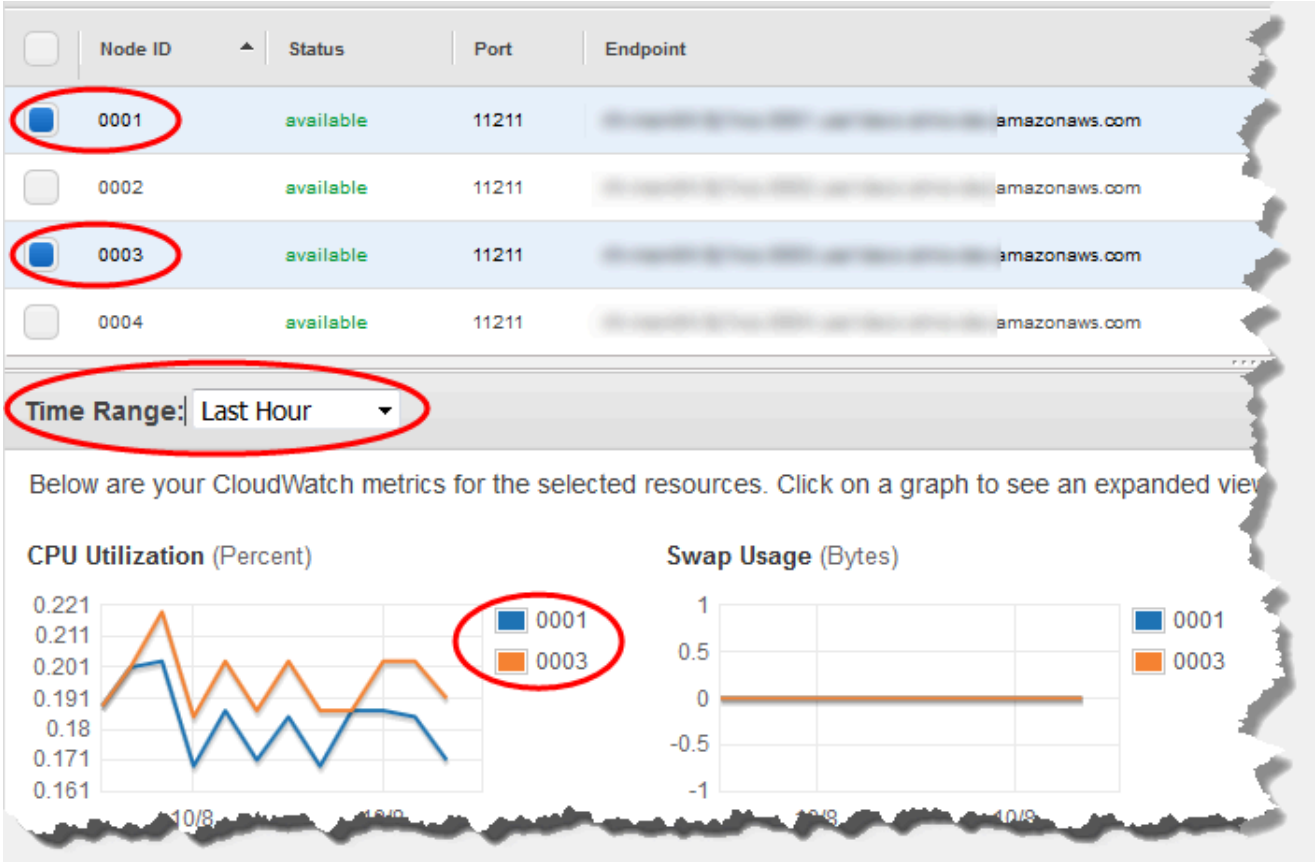
Memcached クラスターの詳細を表示するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/elasticache/ で Amazon 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. ElastiCache コンソールダッシュボードで、Memcached を選択します。これにより、任意のバージョンの Memcached 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。
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

    a. クラスターの名前を選択します。

    b. [Nodes] タブを選択します。

    c. 1 つまたは複数のノードのメトリクスを表示するには、ノード ID の左側のボックスを選択し、[Time range] からメトリクスの時間範囲を選択します。複数のノードを選択すると、オーバーレイグラフが生成されます。



2 つの Memcached ノードに関する過去 1 時間のメトリクス

クラスターの詳細の表示 (AWS CLI)

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

次のコードは my-cluster の詳細を一覧します。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
aws elasticache describe-cache-clusters --max-items 25
```

Example

Linux、macOS、Unix の場合:

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

Windows の場合:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id my-cluster ^
    --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": "my-cluster.7ef-
example.0001.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0002.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
```

```
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0003",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0003.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "my-cluster",
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "my-cluster.7ef-example.cfg.usw2.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
            "CacheNodeType": "cache.m3.medium"
```

```
        }
    ]
}
```

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

クラスターの詳細の表示 (ElastiCache API)

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

次のコードは my-cluster の詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterId=my-cluster
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

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

ElastiCache クラスターの変更

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

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

クラスターのパラメータを変更すると、変更はすぐにクラスターに適用されるか、クラスターが再起動されてから適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。特定のパラメータの変更が適用されるタイミングを確認するには、Memcached 固有のパラメータ および 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。

の使用 AWS Management Console

クラスターを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/elasticache/ で ElastiCache コンソールを開きます。
2. 右上隅のリストから、変更するクラスターがある AWS リージョンを選択します。
3. ナビゲーションペインで、変更するクラスターで実行されているエンジンを選択します。

   選択したエンジンのクラスターが一覧表示されます。
4. クラスターのリストで、変更するクラスターの名前を選択します。
5. アクション を選択してから、変更 を選択します。

   [Modify Cluster] ウィンドウが表示されます。
6. [クラスターの変更] ウィンドウで、必要な変更を行います。オプションには以下が含まれます。

   - エンジンバージョンの互換性
   - VPC セキュリティグループ
   - 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。

7. [変更] を選択します。

の使用 AWS CLI

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

> **Important**
>
> 最新のエンジンバージョンにアップグレードできます。その設定方法の詳細については、「エンジンバージョンとアップグレード」を参照してください。ただし、既存のクラスターを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

`--apply-immediately` パラメータは、エンジンバージョンの変更および 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、「 の ElastiCache トピック AWS CLI 」を参照してくださいmodify-cache-cluster。

ElastiCache API の使用

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

> **Important**
>
> 最新のエンジンバージョンにアップグレードできます。その設定方法の詳細については、「エンジンバージョンとアップグレード」を参照してください。ただし、既存のクラスターを削除して作成し直さない限り、以前のバージョンのエンジンにはダウングレードできません。

改行は読みやすくするために追加しています。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150901T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150901T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

ApplyImmediately パラメータは、ノードタイプとエンジンバージョンの変更、 クラスターのノード数の変更にのみ適用されます。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、 ElastiCache 「 API リファレンストピック」を参照してくださいModifyCacheCluster。

クラスターの再起動

変更内容によっては、変更を適用するためにクラスター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。

クラスターを再起動すると、クラスターのすべてのデータがフラッシュされ、エンジンが再起動されます。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、ElastiCache の AWS CLI、または ElastiCache API を使用して、クラスターを再起動できます。ElastiCache コンソール、AWS CLI、または ElastiCache API を使用する場合でも、再起動を開始できるのは 1 つのクラスターだけです。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。

AWS Management Consoleの使用

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. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 右上隅にある一覧から、目的の AWS リージョンを選択します。
3. ナビゲーションペインで、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タンを選択して、再起動するクラスターを選択します。

   [アクション]、[再起動] の順に選択します。

   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。

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。

AWS CLIの使用

クラスター (AWS CLI) を再起動するには、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、「reboot-cache-cluster」を参照してください。

ElastiCache API の使用

ElastiCache API を使用してクラスターを再起動するには、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、「RebootCacheCluster」を参照してください。

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、「効率的な負荷分散のための ElastiCache クライアントの設定」を参照してください。

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

の使用 AWS Management Console

トピック

- ノードをクラスターに追加するには (コンソール)

ノードをクラスターに追加するには (コンソール)

次の手順を使用して、クラスターにノードを追加できます。

1. 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。
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。

の使用 AWS CLI

を使用してクラスターにノードを追加するには AWS CLI、以下のパラメータ`modify-cache-cluster`を指定して 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少ない場合、 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 を想定します。詳細については、「AWS CLIの使用」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",

    }
}
```

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。

ElastiCache API の使用

クラスターにノードを追加するには (ElastiCache API)

- 以下のパラメーターを指定して ModifyCacheCluster API 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少ない場合、 はクラスターから削除するノードのリストCacheNodeIdsToRemoveを含む パラメータを想定します (ElastiCache 「」を参照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、 ElastiCache 「 API トピック」を参照してくださいModifyCacheCluster。

クラスターからのノードの削除

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、「効率的な負荷分散のための ElastiCache クライアントの設定」を参照してください。

AWS Management Console、AWS CLI、または ElastiCache API を使用して、クラスターからノードを削除できます。

AWS Management Consoleの使用

クラスターからノードを削除するには (コンソール)

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 右上にあるリストから、ノードを削除するクラスターの AWS リージョンを選択します。
3. ナビゲーションペインで、ノードを削除するクラスターで実行されているエンジンを選択します。

   選択したエンジンを実行しているクラスターが一覧表示されます。
4. クラスターの一覧から、ノードを削除するクラスターの名前を選択します。

   クラスターのノードが一覧表示されます。
5. 削除するノードのノード ID の左側にあるボックスをオンにします。ElastiCache コンソール で同時に削除できるノードは 1 つのみです。複数のノードを選択すると、[ノードの削除] ボタンは使用できません。

   [ノードの削除] ページが表示されます。
6. ノードを削除するには、[ノードの削除] ページに入力し、[ノードの削除] を選択します。ノードを保持するには、[キャンセル] を選択します。

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。

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
			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の使用

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

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

   - `--cache-cluster-id` ノードを削除する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。
   - `--cache-node-ids-to-remove` このクラスターから削除するノード ID のリスト。

- `--apply-immediately` または `--no-apply-immediately` は、次のメンテナンス時にこれらのノードを削除するかどうかを指定します。
- `--region` 複数のノードを削除するクラスターの AWS リージョンを指定します。

次の例では、my-cluster クラスターからノード 0001 を直ちに削除します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --num-cache-nodes 2 \
    --cache-node-ids-to-remove 0001 \
    --region us-east-2 \
    --apply-immediately
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --num-cache-nodes 2 ^
    --cache-node-ids-to-remove 0001 ^
    --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)。

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "my-cluster",
        "PreferredAvailabilityZone": "us-east-2b",
        "ConfigurationEndpoint": {
            "Port": 11211,
            "Address": "rlh-mem000.7ef-example.cfg.usw2.cache.amazonaws.com"
        },
```

```
        "CacheSecurityGroups": [],
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z", 9dcv5r
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 3,
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 2,
            "CacheNodeIdsToRemove": [
                "0001"
            ]
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",

    }
}
```

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

ElastiCache API の使用

ElastiCache API を使用してノードを削除するには、次のようにキャッシュクラスター ID と削除するノードの一覧を使用して ModifyCacheCluster API 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するノード ID の一覧。
  - 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 1 つのノードを削除するクラスターの AWS リージョンを指定します。

次の例では、my-cluster クラスターからノード 0004 と 0005 を直ちに削除します。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=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、「ElastiCache API トピック ModifyCacheCluster」を参照してください。

保留中のノードの追加または削除オペレーションのキャンセル

変更を直ちに適用しないことを選択した場合、操作は、次のメンテナンス時間に実行されるまで pending ステータスになります。保留中のオペレーションはすべてキャンセルできます。

保留中のオペレーションをキャンセルするには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 右上のリストから、保留中のノードの追加または削除オペレーションをキャンセルする AWS リージョンを選択します。
3. ナビゲーションペインで、保留中のオペレーションをキャンセルするクラスターで実行されているエンジンを選択します。選択したエンジンを実行しているクラスターが一覧表示されます。
4. クラスターのリストで、キャンセルする保留中のオペレーションがあるクラスターの名前 (クラスター名の左にあるボックスではなく) を選択します。
5. 保留中のオペレーションを確認するには、[説明] タブを選択し、表示されている保留中の作成または削除の数を確認します。保留中の作成と保留中の削除の両方を持つことはできません。
6. [Nodes] タブを選択します。
7. すべての保留中のオペレーションをキャンセルするには、[Cancel Pending] をクリックします。[Cancel Pending] ダイアログボックスが表示されます。
8. [Cancel Pending] ボタンを選択して、すべての保留中のオペレーションをキャンセルすることを確認します。オペレーションを保持する場合は、[Cancel] を選択します。

クラスターの削除

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

AWS Management Consoleの使用

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

クラスターを削除するには

1. AWS Management Console にサインインして、Amazon ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ElastiCache コンソールのダッシュボードで、削除するクラスターで実行されているエンジンを選択します。

   そのエンジンを実行しているすべてのクラスターが一覧表示されます。
3. 削除するクラスターを選択するには、クラスターのリストからそのクラスターの名前を選択します。

   > **Important**
   >
   > ElastiCache コンソールから、一度に 1 つずつクラスターを削除できます。複数のクラスターを選択すると、削除オペレーションが無効になります。
4. アクションとして、Delete (削除) を選択します。
5. [クラスターの削除] 確認画面で、[削除] を選択してクラスターを削除するか、[キャンセル] を選択してクラスターを保持します。

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

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

AWS CLIの使用

次のコードでは、キャッシュクラスター `my-cluster` を削除します。

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

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

Linux、macOS、Unix の場合:

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

Windows の場合:

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

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

ElastiCache API の使用

次のコードはクラスター my-cluster を削除します。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheCluster
    &CacheClusterId=my-cluster
    &Region us-east-2
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150202T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

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

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

クラスターへのアクセス

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

Amazon Virtual Private Cloud (Amazon VPC) で ElastiCache インスタンスを起動した場合、同じ Amazon の Amazon インスタンスからEC2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

- クラスターへのアクセスの許可

クラスターへのアクセスの許可

クラスターを EC2- で起動しました。VPC

クラスターを Amazon Virtual Private Cloud (Amazon VPC) で起動した場合、同じ Amazon で実行されている Amazon EC2インスタンスからのみクラスターに接続 ElastiCacheできます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のサブネット、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 使用するセキュリティグループを作成しない限り、このセキュリティグループにはデフォルトの という名前が付けられます。
4. 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** です。
   e. ソースボックスで、ポート範囲 (0.0.0.0/0) を持つ Anywhere を選択し、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 に開くと、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、クラスターはインターネットに公開されませんVPC。ただし、デフォルトのセキュリティグループは、お客様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はパブリック IP アドレスを持つ場合があります。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 で起動すると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、クラウドベースのインメモリ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す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) インスタンスから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 ElastiCache からリソースにアクセスするときは、次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。影響は、NATインスタンスを介してアクセスするキャッシュクラスターの数に応じて増加します。

- クライアントからNATインスタンスへのトラフィックは暗号化されません。したがって、NATインスタンス経由で機密データを送信しないようにする必要があります。
- 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方向に行う必要があります。

```
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*
- 信頼済みクライアントの IP アドレスが 198.51.100.27 である。
- NAT インスタンスの Elastic IP アドレスは 203.0.113.73 です。
- NAT インスタンスに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 EC2 NAT AMI にアクセスするには、 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、NAT「 ユーザーガイド」の「インスタンス AWS VPC」を参照してください。
2. キャッシュクラスターとNATインスタンスのセキュリティグループルールを作成します。

   NAT インスタンス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
     - 信頼されたクライアントTCPからNATインスタンスから転送された各キャッシュポートへの接続を許可する 1 つ (11211 ~ 11213)。
     - 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 。

     NAT インスタンス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	11211-11213	198.51.100.27/32
SSH	TCP	22	198.51.100.27/32

   - 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。

     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)

- 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。

  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)

3. ルールを検証します。

   - 信頼できるクライアントがNATインスタンスSSHに対して を実行できることを確認します。
   - 信頼できるクライアントが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. iptables ルールをNATインスタンスに追加します。

   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
```

5. 信頼済みクライアントがクラスターに接続できることを確認します。

   信頼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 と、適切なクラスターノードに対応するクラスターポートに接続する必要があります。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
```

6. iptables 設定を保存します。

   ルールをテストし、検証してから保存します。Red Hat ベースの Linux ディストリビューション (Amazon Linux など) を使用している場合は、次のコマンドを実行します。

```
service iptables save
```

関連トピック

以下のトピックも役に立つ場合があります。

- Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン
- お客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする
- NAT インスタンス
- ElastiCache クライアントの設定
- Amazon VPC NAT インスタンスの高可用性: 例

接続エンドポイントの検索

エンドポイントを使用してアプリケーションがクラスターに接続します。エンドポイントはノードまたはクラスターの一意のアドレスです。

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。

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。

クラスターのエンドポイントの検索 (コンソール)

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、「ノードに接続する」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 を作成することを選択した場合、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)

Amazon AWS CLI の を使用して ElastiCache 、ノードとクラスター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)

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 AWS CLI 、 `describe-cache-clusters` コマンドを使用してクラスターとそのノードのエンドポイントを検出できます。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",

        }
    ]
}
```

Important

Memcached 設定エンドポイントCNAMEの を作成することを選択した場合、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。

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

エンドポイントの検索 (ElastiCache API)

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。

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)

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`アクションで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。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

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 を作成することを選択した場合、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。

ノードの管理

ノードは、Amazon ElastiCache デプロイの最小の構成要素です。これは、安全なネットワークに接続された RAM の固定サイズの断片です。各ノードは、クラスターの作成時または最終変更時に選択されたエンジンを実行します。各ノードはそれぞれ Domain Name Service (DNS) 名とポートを持っています。複数のタイプの ElastiCache ノードがサポートされており、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、「Memcached ノードサイズの選択」を参照してください。

トピック

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 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 ノードのステータスを表示できます。

ElastiCache ノードに指定できるステータス値は、次の表のとおりです。この表には、 ElastiCache ノードに対して課金されるかどうかも示されています。

タイプ	請求対象	説明
available	請求対象	ElastiCache ノードは正常で、使用可能です。
creating	非請求対象	ElastiCache ノードを作成中です。ノードの作成中はノードにアクセスできません。
deleting	非請求対象	ElastiCache ノードは削除中です。
modifying	請求対象	お客様が ElastiCache ノードの変更をリクエストしたため、ノードは変更中です。
updating	請求対象	更新状態は、Amazon ElastiCache ノードの次の 1 つ以上が当てはまることを示します。 • 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。

タイプ	請求対象	説明
		• 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。 • ElastiCache (Redis OSS) パスワードは、 を使用して更新/ローテーションされていますAWS Secrets Manager。
`rebooting cache cluster nodes`	請求対象	顧客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 プロセスにより、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 ノードで次の 1 つ以上が true であることを示します。 • 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 VPC) に存在しなくなります。

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、Amazon ElastiCache ノードで次のいずれかに該当することを示します。 • 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、に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容量 AWS がない場合にも、現行世代のノードを置き換えることによって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。 • 指定した RDB スナップショットを復元できなかった。 • 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 ElastiCache (Redis OSS) ノードのスナップショットを作成しています。

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、https://console.aws.amazon.com/elasticache/ で Amazon ElastiCache コンソールを開きます。
2. ナビゲーションペインで、Redis 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、キャッシュに適用可能なサービス更新のリストを表示できます。

での 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
```

API を使用した ElastiCache ノードステータスの表示

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

ノードに接続する

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

次の例では、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. 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. Amazon EC2 インスタンスで [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 インスタンスのコマンドプロンプトで、以下のコマンドを入力し、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ます。この例に示されているノードのエンドポイントを、使用するノードのエンドポイントに置き換えてください。

```
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

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

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

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

- 汎用:
  - 現行世代:

    M6g ノードタイプ (Memcached エンジンバージョン 1.5.16 以降でのみ使用可能)。

    `cache.m6g.large`, `cache.m6g.xlarge`, `cache.m6g.2xlarge`, `cache.m6g.4xlarge`, `cache.m6g.8xlarge`, `cache.m6g.12xlarge`, `cache.m6g.16xlarge`

    > **Note**
    >
    > 利用可能なリージョンについては、「AWS リージョン別のサポート対象ノードタイプ」を参照してください。

    M5 ノードタイプ:
    `cache.m5.large`、`cache.m5.xlarge`、`cache.m5.2xlarge`、`cache.m5.4xlarge`、`cache.m5.`

    M4 ノードタイプ:
    `cache.m4.large`、`cache.m4.xlarge`、`cache.m4.2xlarge`、`cache.m4.4xlarge`、`cache.m4.`

    T4g ノードタイプ (Memcached エンジンバージョン 1.5.16 以降でのみ使用可能)。

    `cache.t4g.micro`, `cache.t4g.small`, `cache.t4g.medium`

    T3 ノードタイプ: `cache.t3.micro`、`cache.t3.small`、`cache.t3.medium`

    T2 ノードタイプ: `cache.t2.micro`、`cache.t2.small`、`cache.t2.medium`
  - 旧世代: (推奨しません。既存のクラスターは引き続きサポートされますが、これらのタイプでは新しいクラスターの作成はサポートされません。)

T1 ノードタイプ: `cache.t1.micro`

M1 ノードタイプ:
`cache.m1.small`、`cache.m1.medium`、`cache.m1.large`、`cache.m1.xlarge`

M3 ノードタイプ:
`cache.m3.medium`、`cache.m3.large`、`cache.m3.xlarge`、`cache.m3.2xlarge`

- コンピューティングの最適化:
  - 以前の世代: (推奨しません)

    C1 ノードタイプ: `cache.c1.xlarge`
- メモリ最適化:
  - 現行世代:

    (R6g ノードタイプは Memcached エンジンバージョン 1.5.16 以降でのみ使用可能)。

    R6g ノードタイプ:
    `cache.r6g.large`、`cache.r6g.xlarge`、`cache.r6g.2xlarge`、`cache.r6g.4xlarge`、`cache`

    > **Note**
    >
    > 利用可能なリージョンについては、「AWS リージョン別のサポート対象ノードタイプ」を参照してください。

    R5 ノードタイプ:
    `cache.r5.large`、`cache.r5.xlarge`、`cache.r5.2xlarge`、`cache.r5.4xlarge`、`cache.r5.`

    R4 ノードタイプ:
    `cache.r4.large`、`cache.r4.xlarge`、`cache.r4.2xlarge`、`cache.r4.4xlarge`、`cache.r4.`
  - 以前の世代: (推奨しません)

    M2 ノードタイプ: `cache.m2.xlarge`、`cache.m2.2xlarge`、`cache.m2.4xlarge`

    R3 ノードタイプ:
    `cache.r3.large`、`cache.r3.xlarge`、`cache.r3.2xlarge`、`cache.r3.4xlarge`、`cache.r3.`
- ネットワーク最適化:

- 現行世代:

  (C7gn ノードタイプは Memcached エンジンバージョン 1.6.6 以降でのみ使用可能)。

  C7gn ノードタイプ:
  `cache.c7gn.large`、`cache.c7gn.xlarge`、`cache.c7gn.2xlarge`、`cache.c7gn.4xlarge`、`c`

現行世代

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
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

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.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
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

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.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

メモリ最適化

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
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

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.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
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

ネットワーク最適化

インスタンスタイプ	サポートされている最小バージョン	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

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-Standard、および T2-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、CPU パフォーマンスのベースラインレベルを提供し、発生したクレジットが使い果たされるまでいつでも CPU 使用率をバーストすることができます。CPU クレジットは、フル CPU パフォーマンスを 1 分間実現します。

Amazon ElastiCacheの T4g, T3、および T2 ノードは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。ベースラインより上にバーストする場合、ノードは CPU クレジット残高に蓄積されたクレジットを消費します。蓄積されたクレジットで実行中のノードが少ない場合、パフォーマンスは次第にベースラインのパフォーマンスレベルに下がります。この段階的な低下により、蓄積された CPU クレジットバランスがなくなったときに、ノードでパフォーマンスの急激な低下が発生することがありません。詳細については、Amazon EC2 ユーザーガイドの「バースト可能パフォーマンスインスタンスの CPU クレジットおよびベースラインパフォーマンス」を参照してください。

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

1 時間あたりに受け取る CPU クレジット	蓄積可能な最大獲得クレジット*	vCPUs	vCPU 別のベースラインパフォーマンス	メモリ (GiB)	ネットワークパフォーマンス
12	288	2	10%	0.5	最大 5 ギガビット
24	576	2	20%	1.37	最大 5 ギガビット
24	576	2	20%	3.09	最大 5 ギガビット
12	288	2	10%	0.5	最大 5 ギガビット
24	576	2	20%	1.37	最大 5 ギガビット
24	576	2	20%	3.09	最大 5 ギガビット
6	144	1	10%	0.5	低から中程度
12	288	1	20%	1.55	低から中程度
24	576	2	20%	3.22	低から中程度

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

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

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

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

- `CPUCreditUsage`
- `CPUCreditBalance`

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

また、以下の詳細についても注意してください。

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

関連情報

- Amazon ElastiCache 製品の機能と詳細
- Memcached ノードタイプ固有のパラメータ

ノードの置換

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 更新を適用するために 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る代替キャッシュノードは、異なる IP アドレスを持つ 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 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交換ウィンドウを変更する機能は、 ElastiCache 通知にメンテナンスウィンドウが含まれている場合にのみ使用できます。通知にメンテナンスウィンドウが含まれていない場合、交換ウィンドウを変更することはできません。

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

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

手順については、「メンテナンスの管理」を参照してください。

- [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、「クラスターからのノードの削除」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、「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。

ElastiCache のリザーブドノード

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

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

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

リザーブドノードによるコスト管理

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

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

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

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

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

トピック

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

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

スタンダードリザーブドノードサービス

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

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

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

提供しているタイプ

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

一部前払い RI では、リザーブド 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。

従来のリザーブドノードサービス

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

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

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

使用量レベル

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

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

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

従来のリザーブドノードサービス

提供タイプ	前払いコスト	使用料	メリット
重度使用	高	時間当たりの使用料が最も低く、リザーブドノードの使用状況にかかわらず期間全体に適用されます。	リザーブドノードの実行が 3 年間で全体の 79% を超える場合は、全体的なコストを最も抑えることができます。
中度使用	中		

提供タイプ	前払いコスト	使用料	メリット
		ノードの実行時間に応じて使用料が時間単位で発生します。ノードが実行していないときは、時間単位の使用量は発生しません。	作業負荷が一定していない場合、または、それほど頻繁には利用しない (3 年間で全体の 40% を超える) 場合に適しています。
軽度使用	低	ノードの実行時間に応じて使用料が時間単位で発生します。ノードが実行していないときは、時間単位の使用量は発生しません。すべての種類のうち最も高い料金設定ですが、課金されるのは、リザーブドノードを使用しているときに限られます。	常に実行することを計画している場合は、全体的なコストが最も高くなります。まれにしかリザーブドノードを使用しない (3 年間で全期間の約 15% を超える程度) 場合は、このオプションが適しています。
オンデマンド使用 (リザーブドノードなし)	なし	時間単位の使用料が最も高くなります。ノードの実行中は常に適用されます。	時間単位のコストが最も高くなります。

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

リザーブドノードサービスに関する詳細情報の入手

リザーブドノードを購入する前に、使用可能なリザーブドノードサービスに関する情報を入手できます。

以下の例では、AWS Management Console、AWS CLI、および ElastiCache API を使用して、利用できるリザーブドノードサービスの料金表と情報を入手する方法を示します。

トピック

リザーブドノードサービスに関する詳細情報の入手 (コンソール)

AWS Management Console を使用して利用可能なリザーブドクラスターサービスの料金とその他の情報を入手するには、次の手順に従います。

利用可能なリザーブドノードサービスの情報を入手するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ナビゲーションペインで、[リザーブドノード] を選択します。
3. [Purchase Reserved Nodes] (リザーブドノードの購入) を選択します。
4. [Engine] (エンジン) で、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
   - 期間
   - 提供タイプ

   選択後、選択したノードごとのコストと合計コストが [Reservation details] (予約の詳細) に表示されます。
6. これらのノードを購入して料金が発生することを防ぐには、[Cancel] を選択します。

リザーブドノードサービスに関する詳細情報の入手 (AWS CLI)

利用可能なリザーブドノードサービスの料金表とその他の情報を入手するには、コマンドプロンプトで次のコマンドを入力します。

```
aws elasticache describe-reserved-cache-nodes-offerings
```

このオペレーションにより、次のような出力が生成されます (JSON 形式)。

```
{
            "ReservedCacheNodesOfferingId": "0xxxxxxx-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": "0xxxxxxx-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": "0xxxxxxx-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、AWS CLI リファレンスの「describe-reserved-cache-nodes-offerings」を参照してください。

リザーブドノードサービスに関する詳細情報の入手 (ElastiCache API)

利用可能なリザーブドノードサービスの料金表と情報を入手するには、`DescribeReservedCacheNodesOfferings` アクションを呼び出します。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodesOfferings
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「DescribeReservedCacheNodesOfferings」を参照してください。

リザーブドノードの購入

以下の例では、AWS Management Console、AWS CLI、および ElastiCache API を使用してリザーブドノードサービスを購入する方法を示します。

> **Important**
>
> このセクションの例に従うと、AWS アカウントで料金が発生します。これを元に戻すことはできません。

トピック

- リザーブドノードの購入 (コンソール)
- リザーブドノードの購入 (AWS CLI)
- リザーブドノードの購入 (ElastiCache API)

リザーブドノードの購入 (コンソール)

この例では、リザーブドノード ID が myreservationID の特定のリザーブドノードサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

次の手順では、AWS Management Console を使用して、提供 ID 別にリザーブドノードサービスを購入します。

リザーブドノードを購入するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ナビゲーションリストで、[Reserved Nodes] (リザーブドノード) リンクを選択します。
3. [Purchase reserved nodes] (リザーブドノードを購入) ボタンを選択します。
4. [Engine] (エンジン) で、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
   - 期間
   - 提供タイプ
   - オプションの [Reserved node ID] (リザーブドノード ID)

選択後、選択したノードごとのコストと合計コストが [Reservation details] (予約の詳細) に表示されます。

6. [Purchase] (購入) を選択します。

リザーブドノードの購入 (AWS CLI)

以下の例では、リザーブドノード ID が myreservationID の特定のリザーブドクラスターサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

コマンドプロンプトで以下のコマンドを入力します。

Linux、macOS、Unix の場合:

```
aws elasticache purchase-reserved-cache-nodes-offering \
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-cache-node-id myreservationID
```

Windows の場合:

```
aws elasticache purchase-reserved-cache-nodes-offering ^
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-cache-node-id myreservationID
```

このコマンドにより、以下のような出力が返されます。

```
RESERVATION  ReservationId      Class           Start Time                Duration
 Fixed Price  Usage Price  Count  State           Description      Offering Type
RESERVATION  myreservationid    cache.xx.small  2013-12-19T00:30:23.247Z  1y
 XXX.XX USD   X.XXX USD    1      payment-pending  memcached        Medium Utilization
```

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

リザーブドノードの購入 (ElastiCache API)

以下の例では、リザーブドクラスター ID が myreservationID の特定のリザーブドノードサービス 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f を購入する方法を示しています。

以下のパラメーターを指定して PurchaseReservedCacheNodesOffering オペレーションを呼び出します。

- ReservedCacheNodesOfferingId = 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
- ReservedCacheNodeID = myreservationID
- CacheNodeCount = 1

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
    &ReservedCacheNodeID=myreservationID
    &CacheNodeCount=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「PurchaseReservedCacheNodesOffering」を参照してください。

リザーブドノードに関する詳細情報の入手

購入したリザーブドノードに関する情報は、AWS Management Console、AWS CLI、および ElastiCache API を使用して入手できます。

トピック

- リザーブドノードに関する詳細情報の入手 (コンソール)
- リザーブドノードに関する詳細情報の入手 (AWS CLI)
- リザーブドノードに関する詳細情報の入手 (ElastiCache API)

リザーブドノードに関する詳細情報の入手 (コンソール)

次の手順では、AWS Management Console を使用して、購入したリザーブドノードに関する情報を入手します。

購入したリザーブドノードに関する情報を入手するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ナビゲーションリストで、[Reserved nodes] (リザーブドノード) リンクを選択します。

   アカウントのリザーブドノードが [Reserved nodes] (リザーブドノード) の一覧に表示されます。リスト内のいずれかのリザーブドノードを選択して、コンソールの下部にある詳細ペインにリザーブドノードの詳細情報を表示できます。

リザーブドノードに関する詳細情報の入手 (AWS CLI)

AWS アカウントのリザーブドノードに関する情報を入手するには、コマンドプロンプトで次のコマンドを入力します。

```
aws elasticache describe-reserved-cache-nodes
```

このオペレーションにより、次のような出力が生成されます (JSON 形式)。

```
{
    "ReservedCacheNodeId": "myreservationid",
    "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
    "CacheNodeType": "cache.xx.small",
```

```
    "Duration": "31536000",
    "ProductDescription": "memcached",
    "OfferingType": "Medium Utilization",
    "MaxRecords": 0
}
```

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

リザーブドノードに関する詳細情報の入手 (ElastiCache API)

AWS アカウントのリザーブドノードに関する情報を取得するには、DescribeReservedCacheNodes オペレーションを呼び出します。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodes
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、ElastiCache API リファレンスの「DescribeReservedCacheNodes」を参照してください。

以前の世代のノードの移行

以前の世代のノードとは、段階的に廃止されるノードタイプです。旧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で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旧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。

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常に空から開始され、アプリケーションに入力できます。

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)」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、「クラスターの削除」を参照してください。

の使用 ElastiCache

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

トピック

スナップショットおよび復元

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。いつでも、新しい Serverless Memcached キャッシュを作成し、バックアップからのデータを入力してデータを復元できます。では ElastiCache、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、および ElastiCache API を使用してバックアップを管理できます。

キャッシュを削除する予定であり、データを保持することが重要な場合は、万一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、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。
- 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。
- バックアッププロセス中に、サーバーレスキャッシュで他の API または CLI オペレーションを実行することはできません。

自動バックアップのスケジュール

Memcached サーバーレスキャッシュの自動バックアップを有効にできます。自動バックアップが有効になっている場合、 はキャッシュのバックアップを毎日 ElastiCache 作成します。キャッシュへの影響はなく、変更は即時に行われます。自動バックアップは、データ損失を防ぐのに役立ちます。障害が起こった場合、最新のバックアップからデータを復元して新しいキャッシュを作成できます。その結果、データがプリロードされたキャッシュがウォームスタートされ、使用可能になります。詳細については、「バックアップから新しいキャッシュへの復元」を参照してください。

自動バックアップをスケジュールすると、 ElastiCache はいつ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、は自動的にバックアップウィンドウを ElastiCache 割り当てます。

ElastiCache コンソール、、または ElastiCache API を使用して、Memcached Serverless キャッシュの作成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。これは、「高度な 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することによって行われます。

手動バックアップの取得

自動バックアップに加えて、いつでも手動バックアップを作成できます。指定された保持期間後に自動的に削除される自動バックアップとは異なり、手動バックアップには、経過した後で自動的に削除される保持期間はありません。キャッシュを削除しても、そのキャッシュからの手動バックアップはすべて保持されます。手動バックアップを保持する必要がなくなった場合は、自分で明示的に削除する必要があります。

手動バックアップの直接的な作成に加えて、以下のいずれかの方法で手動バックアップを作成できます。

- 「バックアップのコピー」ソースのバックアップが自動で作成されたか、手動で作成されたかは問題ではありません。
- 「最終バックアップの作成」クラスターまたはノードを削除する直前に最終バックアップを作成します。

キャッシュの手動バックアップは、 AWS Management Console、、 AWS CLIまたは ElastiCache API を使用して作成できます。

手動バックアップの作成

キャッシュのバックアップを作成するには (コンソール)

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. ナビゲーションペインから、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、以下のパラメータを指定して `create-serverless-snapshot` AWS CLI オペレーションを使用します。

- `--serverless-cache-name` — バックアップするサーバーレスキャッシュの名前。
- `--serverless-cache-snapshot-name` – 作成するスナップショットの名前。

Linux、macOS、Unix の場合:

```
aws elasticache create-serverless-snapshot \
                        --serverless-cache-name CacheName \
                        --serverless-cache-snapshot-name bkup-20231127
```

Windows の場合:

```
aws elasticache create-serverless-snapshot ^
    --serverless-cache-name CacheName ^
    --serverless-cache-snapshot-name bkup-20231127
```

関連トピック

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

最終バックアップの作成

コンソール、 ElastiCache 、または ElastiCache API を使用して AWS CLI最終バックアップを作成できます。

最終バックアップの作成 (コンソール)

ElastiCache コンソールを使用して Memcached サーバーレス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。

キャッシュの削除時に最終バックアップを作成するには、削除ダイアログボックスの「バックアップの作成」で「はい」を選択し、バックアップに名前を付けます。

関連トピック

- AWS Management Consoleの使用

最終バックアップの作成 (AWS CLI)

を使用してキャッシュを削除するときに、最終バックアップを作成できます AWS CLI。

トピック

- サーバーレスキャッシュを削除する場合

サーバーレスキャッシュを削除する場合

最終バックアップを作成するには、以下のパラメータを指定して `delete-serverless-cache` AWS CLI オペレーションを使用します。

- `--serverless-cache-name` – 削除するキャッシュの名前。
- `--final-snapshot-name` – バックアップの名前。

以下のコードは、最終バックアップ `bkup-20231127-final` をキャッシュ `myserverlesscache` の削除時に作成します。

Linux、macOS、Unix の場合:

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

```
        --final-snapshot-name bkup-20231127-final
```

Windows の場合:

```
aws elasticache delete-serverless-cache ^
        --serverless-cache-name myserverlesscache ^
        --final-snapshot-name bkup-20231127-final
```

詳細については、 コマンドリファレンスdelete-serverless-cache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必要に応じて特定のバックアップの詳細を表示するには、`describe-serverless-cache-snapshots` CLI オペレーションを使用します。

例

以下のオペレーションでは、パラメータ `--max-records` を使用して、アカウントに関連付けられた最大 20 個のバックアップをリスト表示します。パラメータ `--max-records` を省略すると、最大 50 個のバックアップが一覧表示されます。

```
aws elasticache describe-serverless-cache-snapshots --max-records 20
```

以下のオペレーションでは、パラメータ `--serverless-cache-name` を使用して、キャッシュ `my-cache` に関連付けられたバックアップのみをリスト表示します。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-name my-cache
```

以下のオペレーションでは、パラメータ `--serverless-cache-snapshot-name` を使用して、バックアップ `my-backup` の詳細を表示します。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-snapshot-
name my-backup
```

詳細については、 AWS CLI コマンドリファレンスの「describe-serverless-cache-snapshots」を参照してください。

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。

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。

バックアップから新しいキャッシュへの復元

既存のバックアップを新しいサーバーレスキャッシュに復元できます。

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)

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。

サーバーレスキャッシュへのバックアップの復元 (AWS CLI)

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:

Windows の場合:

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine memcached
    --snapshot-arns-to-restore Snapshot-ARN
```

Windows の場合:

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

```
    --snapshot-arns-to-restore Snapshot-ARN
```

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作成されたか手動で作成されたかにかかわらず、いつでもバックアップを削除できる削除 API オペレーションを提供します。(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)。

コンソール、 ElastiCache 、または ElastiCache API を使用して AWS CLIバックアップを削除できます。

バックアップの削除 (コンソール)

次の手順では、コンソールを使用してバックアップを削除します ElastiCache 。

バックアップを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/elasticache/ で ElastiCache コンソールを開きます。
2. ナビゲーションペインで、[バックアップ] を選択します。

   [バックアップ] 画面にバックアップのリストが表示されます。
3. 削除するバックアップの名前の左にあるチェックボックスをオンにします。
4. [削除] をクリックします。
5. このバックアップを削除する場合は、[バックアップの削除] 確認画面で [削除] を選択します。ステータスが deleting に変わります。

サーバーレスバックアップの削除 (AWS CLI)

サーバーレスバックアップを削除するには、次のパラメータを指定して delete-snapshot AWS CLI オペレーションを使用します。

- `--serverless-cache-snapshot-name` – 削除するバックアップの名前。

以下のコードはバックアップ `myBackup` を削除します。

```
aws elasticache delete-serverless-cache-snapshot --serverless-cache-snapshot-
name myBackup
```

詳細については、AWS CLI コマンドリファレンスの「delete-serverless-cache-snapshot」を参照してください。

バックアップへのタグ付け

タグ形式で各バックアップに独自のメタデータを割り当てることができます。タグを使用すると、バックアップを用途、所有者、環境などのさまざまな方法で分類できます。これは、同じタイプのリソースが多数ある場合に役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。詳細については、「タグを付けることができるリソース」を参照してください。

コスト配分タグは、コストをタグ値別に請求書にグループ化することで、複数の AWS 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、またはコピーできます。詳細については、「コスト配分タグによるコストのモニタリング」を参照してください。

エンジンバージョンとアップグレード

このセクションでは、サポートされる Memcached エンジンのバージョンとアップグレードする方法について説明します。

トピック

サポートされている ElastiCache (Memcached) バージョン

ElastiCache では、以下の 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 バージョンの

- 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) では、Memcached バージョン 1.6.6 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、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 に 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 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 機能をサポートしています。

- 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。
- メモリをバックグラウンドで回復するための LRU クローラ。
- `--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 の改善には、次のものが含まれます。

- ASCII マルチ取得、CVE-2017-9951、`metadumper` の制限クロールなどの累積的修正。
- 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 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 for 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 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 LRU (least recently used) の管理
- 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初期エンジンおよびバージョンでした。

エンジンバージョンとアップグレード

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機能以外の変更用です。

> **Note**
>
> 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にアップグレードされ、次にプライマリリージョンにアップグレードされます。

ElastiCache Serverless のバージョン管理

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、最新の 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 はコンソールで通知を送信し、 で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、キャッシュを最新のメジャーバージョンにアップグレードできます。

独自設計型 ElastiCache クラスターのバージョン管理

独自設計型 ElastiCache クラスターを使用する場合、キャッシュクラスターを使用するソフトウェアを、 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す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を、 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するかどうか、およびいつアップグレードす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。

新しい Memcached 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て使用する新しいエンジンのバージョンを指定します。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、「クラスターの管理」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。ElastiCache サーバーレスには適用されません。

独自設計型クラスターをアップグレードする際は、以下を考慮してください。

- エンジンのバージョニングは、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、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 予約します。
- 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

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、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。 コンソールを使用してクラスターを変更します。
- を使用するには AWS CLI、「 を使用したクラスターの変更CLI」を参照してください。
- を使用するには ElastiCache API、「 を使用したクラスターの変更API」を参照してください。

エンジンバージョンのアップグレード方法

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、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 ElastiCache API を使用します。

- を使用するには AWS Management Console、「」~「」を参照してくださいの使用 AWS Management Console。
- を使用するには、 AWS CLI「」を参照してくださいの使用 AWS CLI。
- ElastiCache API を使用するには、「」を参照してください ElastiCache API の使用。

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

- Memcached クライアントに関するベストプラクティス
- サポートされている Memcached のコマンド
- キャッシュ戦略

Memcached クライアントに関するベストプラクティス

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の詳細については、以下のトピックを参照してください。

トピック

- 効率的な負荷分散のための ElastiCache クライアントの設定
- IPv6 クライアントの例

効率的な負荷分散のための ElastiCache クライアントの設定

> **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、ノード間でキャッシュキーを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 .NET を使用している場合は、いずれかの Amazon ElastiCache クライアントライブラリ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、整合性のあるハッシュ機能が組み込まれたオープンソースの spymemcached Java クライアントに基づいています。このライブラリには、整合性のある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、KetamaConnectionFactory で KetamaConnectionFactory ドキュメントを参照してください。

PHP を使用した整合性のあるハッシュ

ElastiCache Memcached PHP クライアントは、組み込みの Memcached PHP ライブラリのラッパーです。デフォルトでは、整合性のあるハッシュは Memcached PHP ライブラリによって無効になっています。

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、Memcached PHP の実行時設定に関するドキュメント (http://php.net/manual/en/memcached.configuration.php) 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。

.NET を使用した整合性のあるハッシュ

ElastiCache Memcached .NET クライアントは、Enyim Memcached 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、`memcached/locator` のドキュメント (https://github.com/enyim/EnyimMemcached/wiki/MemcachedClient-Configuration#user-content-memcachedlocator) を参照してください。

IPv6 クライアントの例

> **Note**
>
> このセクションは、独自設計型 Memcached クラスターに適用されます。

ElastiCache はオープンソースの Memcached と互換性があります。つまり、IPv6 接続をサポートする Memcached のオープンソースクライアントは、IPv6 対応 ElastiCache (Memcached) クラスターに接続できる必要があります。さらに、以下のクライアントは、サポートされているすべてのネットワークタイプ設定で動作することが具体的に検証されています。

一般的に使用されるオープンソースのクライアントライブラリで IPv6 対応 ElastiCache リソースを操作するためのベストプラクティスを次に示します。 ElastiCache リソースのクライアント設定に関

する推奨事項については、 を操作するための既存のベストプラクティス ElastiCacheを参照してください。ただし、IPv6 対応リソースを操作する際には、注意すべき点がいくつかあります。

検証済みクライアント

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 クラスターでは、IP 検出パラメータを使用して、クライアントがクラスター内のノードに接続するために使用するプロトコルを制御できます。IP 検出パラメータは、IPv4 または IPv6 に設定できます。

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で、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の IP が報告されると、変更の反映が完了します。

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 対応デュアルスタック ElastiCache クラスター

ElastiCache クラスターで TLS が有効になっている場合、クラスター検出関数 は IP の代わりに`config get cluster`ホスト名を返します。 IPs その後、ホスト名は IPs の代わりに ElastiCache クラスターに接続し、TLS ハンドシェイクを実行します。つまり、クライアントは IP

検出パラメータの影響を受けません。TLS が有効なクラスターでは、IP 検出パラメータは優先 IP プロトコルに影響しません。代わりに、使用する IP プロトコルは、DNS ホスト名を解決する際にクライアントがどの IP プロトコルを使用するかによって決まります。

Java クライアント

IPv4 と IPv6 の両方をサポートする Java 環境から接続する場合、後方互換性のために Java はデフォルトで IPv6 よりも IPv4 を優先します。ただし、IP プロトコルプリファレンスは JVM 引数を使用して設定できます。IPv4 を優先するには、JVM は `-Djava.net.preferIPv4Stack=true` を受け入れ、IPv6 セット `-Djava.net.preferIPv6Stack=true` を優先します。`-Djava.net.preferIPv4Stack=true` を設定すると、JVM は IPv6 接続を行わなくなります。

ホストレベルの設定

一般に、クライアントまたはクライアントランタイムに IP プロトコルプリファレンスを設定するための構成オプションが提供されていない場合、DNS 解決を実行するとき、IP プロトコルはホストの設定に依存します。デフォルトでは、ほとんどのホストは IPv4 よりも IPv6 を優先しますが、この優先度はホストレベルで設定できます。これは、 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時に IPv4 を優先するように設定を更新するには、クラスター IPs を含む CIDR 範囲の優先順位をデフォルトの IPv6 接続の優先順位よりも高く更新します。デフォルトでは、IPv6 接続の優先順位は 40 です。例えば、クラスターが CIDR 172.31.0.0:0/16 のサブネットにあると仮定すると、以下の設定では、クライアントはそのクラスターへの IPv4 接続を優先することになります。

```
label ::1/128       0
label ::/0          1
label 2002::/16     2
label ::/96         3
label ::ffff:0:0/96 4
label fec0::/10     5
label fc00::/7      6
label 2001:0::/32   7
label ::ffff:172.31.0.0/112 8
```

```
#
#    This default differs from the tables given in RFC 3484 by handling
#    (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
#    The reason for this difference is that these addresses are never
#    NATed while IPv4 site-local addresses most probably are.  Given
#    the precedence of IPv6 over IPv4 (see below) on machines having only
#    site-local IPv4 and IPv6 addresses a lookup for a global address would
#    see the IPv6 be preferred.  The result is a long delay because the
#    site-local IPv6 addresses cannot be used while the IPv4 address is
#    (at least for the foreseeable future) NATed.  We also treat Teredo
#    tunnels special.
#
# precedence  <mask>   <value>
#    Add another rule to the RFC 3484 precedence table.  See section 2.1
#    and 10.3 in RFC 3484.  The default is:
#
precedence  ::1/128       50
precedence  ::/0          40
precedence  2002::/16     30
precedence ::/96          20
precedence ::ffff:0:0/96  10
precedence ::ffff:172.31.0.0/112 100
```

gai.conf の詳細については、Linux のメインページを参照してください。

Windows ホスト

Windows ホストのプロセスも同様です。Windows ホストの場合は netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL を実行できます。これは Linux ホストで gai.conf ファイルを変更するのと同じ効果があります。

これにより、指定された CIDR 範囲の IPv6 接続よりも IPv4 接続を優先するように優先設定ポリシーが更新されます。例えば、クラスターが 172.31.0.0:0/16 CIDR のサブネット内にあると仮定した場合、netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15 を実行すると、次の優先順位表になり、クライアントがクラスターに接続する際に IPv4 を優先することになります。

```
C:\Users\Administrator>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence Label Prefix
---------- ----- --------------------------------
```

```
100 15 ::ffff:172.31.0.0:0/112
20 4 ::ffff:0:0/96
50 0 ::1/128
40 1 ::/0
30 2 2002::/16
5 5 2001::/32
3 13 fc00::/7
1 11 fec0::/10
1 12 3ffe::/16
1 3 ::/96
```

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ています。

- クライアント接続には TLS が必要なため、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 および LRU クローラー設定の変更
  - `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

キャッシュ戦略

以下のトピックでは、キャッシュを設定および維持するための戦略について説明します。

キャッシュするデータとデータへのアクセスパターンに基づいて、キャッシュを入力し維持するために実装する戦略とは何か。たとえば、ゲームサイトやトレンドのニュースのランキングトップ 10 で同じ同じ戦略は使用したくないでしょう。このセクションの後半では、一般的なキャッシュのメンテナンス戦略、利点および欠点について説明します。

トピック

遅延読み込み

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) の間にあるインメモリ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 TTL の追加 戦略を使用できます。

遅延読み込み擬似コードの例

次のコードは、遅延読み込みロジックの擬似コードの例です。

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application
// ****************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)
    if (customer_record == null)

        customer_record = db.query("SELECT * FROM Customers WHERE id = {0}",
 customer_id)
        cache.set(customer_id, customer_record)

    return customer_record
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
customer_record = get_customer(12345)
```

書き込みスルー

書き込みスルー戦略では、データがデータベースに書き込まれると常にデータを追加するか、キャッシュのデータを更新します。

書き込みスルーの利点と欠点

書き込みスルーの利点は次のとおりです。

- キャッシュのデータが古くなりません。

  キャッシュにデータベースにデータが書き込まれるたびにキャッシュのデータが更新されるため、キャッシュのデータが常に最新の状態になります。

- 書き込みペナルティ対読み取りペナルティ。

  1 回の書き込みで 2 回のトリップ:

  1. キャッシュへの書き込み
  2. データベースへの書き込み

  レイテンシーをプロセスに追加します。つまり、エンドユーザーは一般的に、データの取得時よりもデータの更新時のレイテンシーに対して寛容です。更新は作業量が大きく時間がかかるのが常です。

書き込みスルーの欠点は次のとおりです。

- 欠落データ。

  ノード障害またはスケールアウトにより、新規ノードをスピンアップすると、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。[有効期限 (TTL) の値を追加する] を使用すると、無駄なスペースを最小限に抑えることができます。

書き込みスルー擬似コードの例

以下は、書き込みスルーロジックの擬似コードの例です。

```
// ****************************************
// function that saves a customer's record.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record)
    return success
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
save_customer(12345,{"address":"123 Main"})
```

TTL の追加

遅延読み取りはデータが古くなる可能性がありますが、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。それぞれの書き込みに有効期限 (TTL) の値を追加することで、それぞれの戦略のメリットが得られます。同時に、過剰なデータでキャッシュがいっぱいになる事態が避けられます。

[有効期限 (TTL)] は、キーの有効期限までの秒数を指定する整数値です。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、 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、TTL のある書き込みスルーロジックの擬似コードの例です。

```
// ****************************************
// function that saves a customer's record.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and future reads will have to query the database.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record, 300)

    return success
```

以下は、TTL のある遅延読み込みロジックの擬似コードの例です。

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
```

```
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and subsequent reads will have to query the database.
// ***************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)

    if (customer_record != null)
        if (customer_record.TTL < 300)
            return customer_record        // return the record and exit function

    // do this only if the record did not exist in the cache OR
    //    the TTL was >= 300, i.e., the record in the cache had expired.
    customer_record = db.query("SELECT * FROM Customers WHERE id = {0}", customer_id)
    cache.set(customer_id, customer_record, 300)  // update the cache
    return customer_record                // return the newly retrieved record and exit
 function
```

この例では、データを取得するアプリケーションコードは次のとおりです。

```
save_customer(12345,{"address":"123 Main"})
```

```
customer_record = get_customer(12345)
```

関連トピック

- インメモリデータストア
- エンジンとバージョンの選択
- スケーリング ElastiCache (Memcached)

独自設計型クラスターの管理

このセクションでは、独自設計型クラスターの管理に役立つトピックを説明します。

> **Note**
>
> これらのトピックは、ElastiCache サーバーレスには適用されません。

トピック

メンテナンスの管理

すべてのクラスターには、週ごとのメンテナンス時間があります。その時間内にシステムの変更が適用されます。クラスターの作成または変更時に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

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-southeast-1	アジアパシフィック (シンガポール) リージョン	14:00 ~ 22:00 UTC
cn-north-1	中国 (北京) リージョン	14:00 ~ 22:00 UTC
cn-northwest-1	中国 (寧夏) リージョン	14:00 ~ 22:00 UTC
ap-east-1	アジアパシフィック (香港) リージョン	13:00 ~ 21:00 UTC
ap-southeast-2	アジアパシフィック (シドニー) リージョン	12:00 ~ 20:00 UTC
eu-west-3	EU (パリ) リージョン	23:59 ~ 07:29 UTC
af-south-1	アフリカ (ケープタウン) リージョン	13:00 ~ 21:00 UTC
eu-central-1	欧州 (フランクフルト) リージョン	23:00 ~ 07:00 UTC
eu-west-1	欧州 (アイルランド) リージョン	22:00 ~ 06:00 UTC
eu-west-2	欧州 (ロンドン) リージョン	23:00 ~ 07:00 UTC
me-south-1	中東 (バーレーン) リージョン	13:00 ~ 21:00 UTC
me-central-1	中東 (アラブ首長国連邦) リージョン	13:00 ~ 21:00 UTC
eu-south-1	欧州 (ミラノ) リージョン	21:00 ~ 05:00 UTC
sa-east-1	南米 (サンパウロ) リージョン	01:00 ~ 09:00 UTC
us-east-1	米国東部 (バージニア北部) リージョン	03:00 ~ 11:00 UTC
us-east-2	米国東部 (オハイオ) リージョン	04:00 ~ 12:00 UTC

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
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

クラスターのメンテナンスウィンドウの変更

メンテナンスウィンドウは使用率の最も低い時間帯に設定する必要があります。このため、場合によっては変更が必要になります。クラスターを変更して、リクエストしたメンテナンス作業が発生するまでの時間範囲 (最大 24 時間) を指定することができます。お客様がリクエストした延期または保留中のクラスターの変更は、この時間に行われます。

> **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。API を使用するには、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 — メンテナンスとノード交換に関するよくある質問
- ノードの置換—ノード交換の管理
- 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

パラメータグループを使用したエンジンパラメータの設定

Amazon ElastiCache はパラメータを使用して、ノードとクラスターの実行時のプロパティを制御します。通常、新しいエンジンバージョンには新しい機能をサポートするための追加のパラメータが含

まれます。パラメータのテーブルについては、「Memcached 固有のパラメータ」を参照してください。

もちろん、maxmemory などのパラメータ値はエンジンやノードのタイプによって決まります。ノードタイプ別のパラメータ値のテーブルについては、「Memcached ノードタイプ固有のパラメータ」を参照してください。

> **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

パラメータの管理

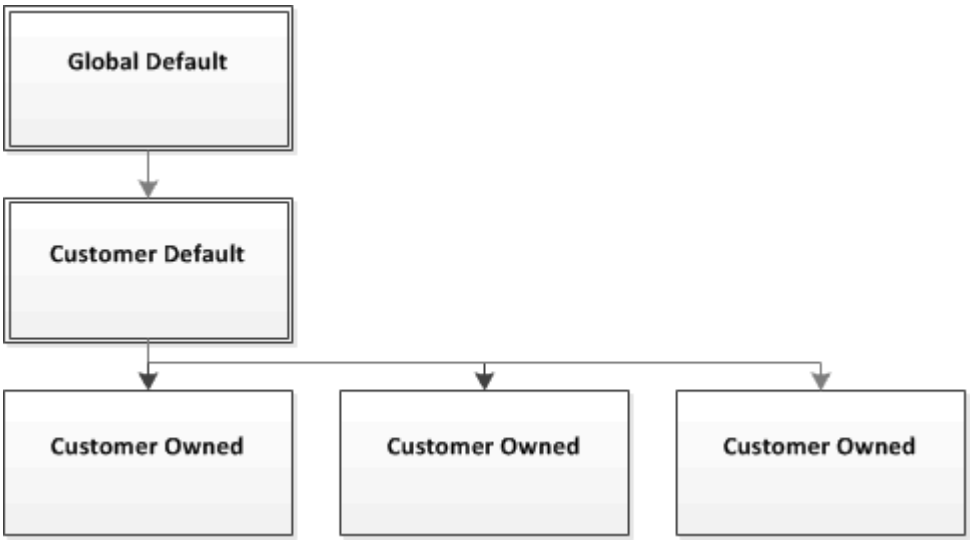
パラメータの管理を容易にするために、パラメータは名前付きのパラメータグループに分類されます。パラメータグループは、起動時にエンジンソフトウェアに渡されるパラメータの特定の値の組み合わせを表しています。これらの値により、各ノードのエンジンプロセスが実行時にどのように動作するかが決まります。特定のパラメータグループのパラメータ値は、クラスターが属するグループに関係なく、そのグループに関連付けられているすべてのノードに適用されます。

クラスターのパフォーマンスを最適化するには、パラメータ値を変更するか、またはクラスターのパラメータグループを変更できます。

- デフォルトのパラメータグループの変更や削除はできません。カスタムパラメータ値が必要な場合は、独自のパラメータグループを作成する必要があります。
- パラメータグループファミリーとユーザーが割り当てているクラスターには、互換性が必要です。たとえば、クラスターが 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。
- クラスターのパラメータグループを変更する場合は、条件付きで変更可能なパラメータの値は、現在のパラメータグループと新しいパラメータグループで一致している必要があります。
- クラスターのパラメータを変更すると、その変更は即座にクラスターに適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。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。

キャッシュパラメータグループの階層

Amazon ElastiCache には、以下に示すキャッシュパラメータグループの 3 つの階層があります。



Amazon ElastiCache パラメータグループの階層

グローバルデフォルト

リージョン内のすべての Amazon ElastiCache のお客様向け最上位ルートパラメータグループ。

グローバルデフォルトのキャッシュパラメータグループ:

- ElastiCache 向けに確保されており、お客様が使用することはできません。

お客様デフォルト

グローバルデフォルトのキャッシュパラメータグループのコピーは、お客様が使用するために作成されています。

お客様デフォルトのキャッシュパラメータグループ:

- ElastiCache によって作成され、所有されています。
- このキャッシュパラメータグループでサポートされているエンジンのバージョンを実行しているすべてのクラスターのキャッシュパラメータグループとして使用できます。
- お客様が編集することはできません。

お客様所有

お客様デフォルトのキャッシュパラメータグループのコピー。お客様所有のキャッシュパラメータグループは、お客様がキャッシュパラメータグループを作成する度に作成されます。

お客様所有のキャッシュパラメータグループ:

- お客様が作成、所有します。
- お客様の互換性のあるいずれのクラスターにも割り当てることができます。
- カスタムキャッシュパラメータグループを作成するようにお客様が変更できます。

  すべてのパラメータ値を変更できるわけではありません。詳細については、「Memcached 固有のパラメータ」を参照してください。

パラメータグループを作成する

デフォルト値から変更するパラメータの値が 1 つ以上ある場合、新しいパラメータグループを作成する必要があります。ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、パラメータグループを作成できます。

パラメータグループを作成する (コンソール)

次の手順では、ElastiCache コンソールを使用してパラメータグループを編集する方法を示します。

ElastiCache コンソールを使用してパラメータグループを作成するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループを作成するには、[Create Parameter Group] を選択します。

   [Create Parameter Group] 画面が表示されます。
4. [Family] のリストから、パラメータグループのテンプレートとなるパラメータグループファミリーを選択します。

   memcached1.4 や などのパラメータグループファミリーは、パラメータグループの実際のパラメータおよびその初期値を定義します。パラメータグループファミリーは、クラスターのエンジンおよびバージョンと一致している必要があります。
5. [Name] ボックスで、このパラメータグループの一意の名前を入力します。

クラスターを作成、またはクラスターのパラメータグループを変更するときは、パラメータグループを名前で選択します。したがって、わかりやすくパラメータグループのファミリーを特定するのに役立つ名前をお勧めします。

パラメータグループの命名に関する制約は次のとおりです。

- 先頭を ASCII 文字にする必要があります。
- ASCII 文字、数字、ハイフンのみを含めることができます。
- 1 ~ 255 文字にする必要があります。
- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。

6. [Description] ボックスに、パラメータグループの説明を入力します。
7. パラメータグループを作成するには、[Create] を選択します。

   パラメータグループを作成しないでプロセスを終了するには、[Cancel] を選択します。
8. パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

パラメータグループを作成する (AWS CLI)

AWS CLI を使用してパラメータグループを作成するには、以下のパラメータを指定して `create-cache-parameter-group` コマンドを使用します。

- `--cache-parameter-group-name` — パラメータグループの名前。

  パラメータグループの命名に関する制約は次のとおりです。
  - 先頭を ASCII 文字にする必要があります。
  - ASCII 文字、数字、ハイフンのみを含めることができます。
  - 1 ~ 255 文字にする必要があります。
  - 連続する 2 つのハイフンを含めることはできません。
  - ハイフンで終わることはできません。
- `--cache-parameter-group-family` — パラメータグループのエンジンとバージョンファミリー。
- `--description` — パラメータグループについてユーザーが入力する説明。

Example

次の例では、memcached1.4 ファミリーをテンプレートとして使用して、myMem14 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"
    }
}
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

詳細については、「create-cache-parameter-group」を参照してください。

パラメータグループを作成する (ElastiCache API)

ElastiCache API を使用してパラメータグループを作成するには、以下のパラメータを指定して CreateCacheParameterGroup アクションを使用します。

- ParameterGroupName — パラメータグループの名前。

パラメータグループの命名に関する制約は次のとおりです。

- 先頭を ASCII 文字にする必要があります。
- ASCII 文字、数字、ハイフンのみを含めることができます。
- 1 ~ 255 文字にする必要があります。
- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。

- `CacheParameterGroupFamily` — パラメータグループのエンジンとバージョンファミリー。例えば、`memcached1.4` です。
- `Description` — パラメータグループについてユーザーが入力する説明。

Example

次の例では、memcached1.4 ファミリーをテンプレートとして使用して、myMem14 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>
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

詳細については、「CreateCacheParameterGroup」を参照してください。

パラメータグループを名前別に一覧表示する

パラメータグループは、 ElastiCache コンソール、、 AWS CLIまたは ElastiCache API を使用して一覧表示できます。

パラメータグループを名前別に一覧表示する (コンソール)

次の手順は、 ElastiCache コンソールを使用してパラメータグループのリストを表示する方法を示しています。

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。

パラメータグループを名前別に一覧表示する (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 のリストです。

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リストには、最大で 10 個のパラメータグループが一覧されています。

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

このコマンドの JSON 出力は、名前、ファミリー、説明を一覧表示し、redis5.6 の場合は、パラメータグループが Global Datastore の一部である (isGlobal) かどうかを各パラメータグループについて表示します。

```
{
    "CacheParameterGroups": [
        {
            "CacheParameterGroupName": "custom-redis32",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "custom parameter group with reserved-memory > 0"
        },
        {
            "CacheParameterGroupName": "default.memcached1.4",
            "CacheParameterGroupFamily": "memcached1.4",
            "Description": "Default parameter group for memcached1.4"
        },
        {
            "CacheParameterGroupName": "default.redis2.6",
            "CacheParameterGroupFamily": "redis2.6",
            "Description": "Default parameter group for redis2.6"
        },
        {
            "CacheParameterGroupName": "default.redis2.8",
            "CacheParameterGroupFamily": "redis2.8",
            "Description": "Default parameter group for redis2.8"
        },
```

```
        {
            "CacheParameterGroupName": "default.redis3.2",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Default parameter group for redis3.2"
        },
        {
            "CacheParameterGroupName": "default.redis3.2.cluster.on",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Customized default parameter group for redis3.2 with
 cluster mode on"
        },
        {
            "CacheParameterGroupName": "default.redis5.6.cluster.on",
            "CacheParameterGroupFamily": "redis5.0",
            "Description": "Customized default parameter group for redis5.6 with
 cluster mode on",
            "isGlobal": "yes"
        },
    ]
}
```

詳細については、「describe-cache-parameter-groups」を参照してください。

パラメータグループを名前で一覧表示する (ElastiCache API)

ElastiCache API を使用してパラメータグループのリストを生成するには、DescribeCacheParameterGroupsアクションを使用します。パラメータグループの名前を指定した場合は、そのパラメータグループのみが一覧表示されます。パラメータグループの名前を指定しない場合は、最大で MaxRecords のパラメータグループが一覧表示されます。いずれの場合も、パラメータグループの名前、ファミリー、および説明が表示されます。

Example

次のサンプルコードは、パラメータグループ myMem14 のリストです。

```
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応答は、名前、ファミリー、説明を一覧表示し、redis5.6 の場合は、パラメータグループが Global Datastore に属している (isGlobal) かどうかを各パラメータグループについて説明します。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
```

```
        <CacheParameterGroupName>myRedis28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 parameter group</Description>
      </CacheParameterGroup>
       <CacheParameterGroup>
        <CacheParameterGroupName>myRedis56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

詳細については、「DescribeCacheParameterGroups」を参照してください。

パラメータグループの値を一覧表示する

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、パラメータグループのパラメータとその値を一覧表示できます。

パラメータグループの値を一覧表示する (コンソール)

次の手順は、ElastiCache コンソールを使用してパラメータグループのパラメータと値を一覧表示する方法を示しています。

ElastiCache コンソールを使用してパラメータグループのパラメータとその値を表示するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループ名の左側にあるボックスを選択して、パラメータと値を一覧表示するパラメータグループを選択します。

   パラメータと値は画面の下部に表示されます。パラメータの数によっては、スクロールして関心のあるパラメータを検索するに必要がある場合もあります。

パラメータグループの値を一覧表示する (AWS CLI)

AWS CLI を使用してパラメータグループのパラメータとその値の一覧を表示するには、`describe-cache-parameters` コマンドを使用します。

Example

次のサンプルコードは、パラメータグループ myMem14 のすべてのパラメータと値リストを一覧します。

Linux、macOS、Unix の場合:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myMem14
```

Windows の場合:

```
aws elasticache describe-cache-parameters ^
```

```
    --cache-parameter-group-name myMem14
```

詳細については、「describe-cache-parameters」を参照してください。

パラメータグループの値を一覧表示する (ElastiCache API)

ElastiCacheAPI を使用してパラメータグループのパラメータとその値を一覧表示するには、DescribeCacheParameters アクションを使用します。

Example

次のサンプルコードは、パラメータグループ myMem14 のすべてのパラメータを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=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>
```

詳細については、「DescribeCacheParameters」を参照してください。

パラメータグループを変更する

> **Important**
>
> デフォルトのパラメータグループを変更することはできません。

パラメータグループでいくつかのパラメータを変更できます。これらのパラメータ値は、パラメータグループに関連付けられるクラスターに適用されます。パラメータ値の変更がパラメータグループに適用される場合の詳細については、「Memcached 固有のパラメータ」を参照してください。

パラメータグループを変更する (コンソール)

次の手順は、 ElastiCache コンソールを使用してbinding_protocol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. [パラメータグループの編集] 画面で、`cluster-enabled` 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 `yes` と入力します。
7. [変更の保存] をクリックします。
8. 変更したパラメータの名前を検索するには、「Memcached 固有のパラメータ」を参照してください。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。

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

変更するパラメータの名前と許容値を検索するには、「Memcached 固有のパラメータ」を参照してください。

次のサンプルコードでは、パラメータグループ 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"
}
```

詳細については、「modify-cache-parameter-group」を参照してください。

パラメータグループの変更 (ElastiCache API)

ElastiCache API を使用してパラメータグループのパラメータ値を変更するには、ModifyCacheParameterGroupアクションを使用します。

Example

変更するパラメータの名前と許容値を検索するには、「Memcached 固有のパラメータ」を参照してください。

次のサンプルコードでは、パラメータグループ 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>
```

詳細については、「ModifyCacheParameterGroup」を参照してください。

パラメータグループを削除する

ElastiCache コンソール、AWS CLI、または ElastiCache API を使用して、カスタムパラメータグループを削除できます。

パラメータグループがクラスターに関連付けられている場合は、パラメータグループを削除できません。デフォルトのパラメータグループも削除できません。

パラメータグループを削除する (コンソール)

次の手順では、ElastiCache コンソールを使用してパラメータグループを削除する方法を示します。

ElastiCache コンソールを使用してパラメータグループを削除するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで [Parameter Groups] を選択します。
3. パラメータグループ名の左側にあるボックスを選択して、削除するパラメータグループを選択します。

   [Delete] ボタンがアクティブになります。
4. [Delete] (削除) をクリックします。

   [Delete Parameter Groups] の確認画面が表示されます。
5. パラメータグループを削除するには、[Delete Parameter Groups] の確認画面で [Delete] を選択します。

   パラメータグループを保持するには、[Cancel] を選択します。

パラメータグループを削除する (AWS CLI)

AWS CLI を使用してパラメータグループを削除するには、`delete-cache-parameter-group` コマンドを使用します。削除するパラメータグループで、`--cache-parameter-group-name` で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

次のサンプルコードは、myMem14 パラメータグループを削除します。

Example

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myMem14
```

Windows の場合:

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myMem14
```

詳細については、「delete-cache-parameter-group」を参照してください。

パラメータグループを削除する (ElastiCache API)

ElastiCache API を使用してパラメータグループを削除するには、DeleteCacheParameterGroup アクションを使用します。削除するパラメータグループで、CacheParameterGroupName で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

Example

次のサンプルコードは、myMem14 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>
```

詳細については、「DeleteCacheParameterGroup」を参照してください。

Memcached 固有のパラメータ

Memcached クラスターにパラメータグループを指定しない場合、エンジンのバージョンに適したデフォルトのパラメータグループが使用されます。デフォルトのパラメータグループのパラメータの値を変更することはできません。ただし、カスタムパラメータグループを作成し、いつでもクラスターに割り当てることはできます。詳細については、「パラメータグループを作成する」を参照してください。

トピック

Memcached 1.6.17 の変更点

Memcached 1.6.17 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 、slab_automove 、lru_crawler 、lru_maintainer 、maxconns_fast コマンドを無効にするためのエイリアス。No modern は、hash_algorithm も jenkins に設定し、ASCII VALUE のインラインも許可します。Memcached 1.5 以上に適用されます。最新に戻すには、このパラメータを無効にして再起動する必要があります。これにより、slab_reassign 、slab_automove 、lru_crawler 、lru_maintainer 、および maxconns_fast が自動的に有効になります。 **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 Elasticache の新規ユーザーによって自動的に取得されます。2021 年 8 月 20 日以前のリージョンで ElastiCache を使用する既

名前	詳細	説明
		存ユーザーは、この新しい変更を取得するためにカスタムパラメータグループを手動で変更する必要があります。
inline_ascii_resp	デフォルト: 0 タイプ : ブール値 変更可能: はい 許可された値: 0,1 変更の適用: 起動時	アイテム内の VALUE レスポンスからの数値が保存されます。最大 24 バイトを使用します。ASCII get、faster セットの小さい減速。

Memcached 1.5.10 では、次のパラメータが削除されます。

名前	詳細	説明
expirezero_does_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	このバージョンではサポートされなくなりました。このバージョン以降、起動または再起動するたびに no-modern がデフォルトで有効になります。

名前	詳細	説明
	されている場合は再起動が必要です) 許可された値: 0,1 変更の適用: 起動時	

Memcached 1.4.34 で追加されたパラメータ

Memcached 1.4.34 では、追加のパラメータはサポートされません。

パラメータグループファミリー: memcached1.4

Memcached 1.4.33 で追加されたパラメータ

Memcached 1.4.33 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: memcached1.4

名前	詳細	説明
modern	デフォルト: 有効 タイプ : ブール値 変更可能: はい 変更の適用: 起動時	各種機能のエイリアス 有効化 modern は、次のコマンドをオンにし、murmur3 ハッシュアルゴリズム (slab_reassign 、slab_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	ログ取得、削除または変異。たとえば、watch をオンにすると、get、set、delete または

名前	詳細	説明
	変更可能: はい 変更の適用: 即時 ログは、watcher_logbuf_size および worker_logbuf_size 制限に達すると削除できます。	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
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。

名前	詳細	説明
watcher_logbuf_size	デフォルト: 256 (KB) タイプ: 整数 変更可能: はい 変更の適用: 起動時	watch コマンドは、Memcached の配信ログ作成をオンにします。ただし、削除、変異、取得によって、ロギングバッファがいっぱいになる可能性がある場合には、watch でログを削除することができます。このような場合、ユーザーは、バッファサイズを増やして、ログ損失の可能性を抑えることができます。
worker_logbuf_size	デフォルト: 64 (KB) タイプ: 整数 変更可能: はい 変更の適用: 起動時	watch コマンドは、Memcached の配信ログ作成をオンにします。ただし、削除、変異、取得によって、ロギングバッファがいっぱいになる可能性がある場合には、watch でログを削除することができます。このような場合、ユーザーは、バッファサイズを増やして、ログ損失の可能性を抑えることができます。
slab_chunk_max	デフォルト: 524288 (バイト) タイプ: 整数 変更可能: はい 変更の適用: 起動時	スラブの最大サイズを指定します。スラブサイズを小さくすると、メモリは効率的に使用されます。slab_chunk_max より大きい項目は、複数のスラブに分割されます。

名前	詳細	説明
`lru_crawler metadump [all|1|2|3]`	デフォルト: 無効 タイプ : ブール値 変更可能: はい 変更の適用: 即時	lru_crawler を有効化すると、このコマンドによってすべてのキーがダンプされます。 `all|1|2|3` - すべてのスラブ、または特定のスラブ数を指定する

Memcached 1.4.24 で追加されたパラメータ

Memcached 1.4.24 では、次のパラメータが追加でサポートされます。

パラメータグループファミリー: memcached1.4

名前	詳細	説明
`disable_flush_all`	デフォルト: 0 (無効) タイプ : ブール値 変更可能: はい 変更の適用: 起動時	flush_all を無効化するパラメータ (-F) を追加します。本稼働インスタンスでフルフラッシュを実行しない場合に便利です。 値: 0、1 (値が 0 の場合にユーザーは `flush_all` を実行できます。)
`hash_algorithm`	デフォルト: jenkins タイプ: 文字列 変更可能: はい 変更の適用: 起動時	使用されるハッシュアルゴリズム。使用可能な値: murmur3 と jenkins。
`lru_crawler`	デフォルト: 0 (無効) タイプ : ブール値	期限が切れた項目のスラブクラスを消去します。これにより、バックグラウンドで実行されるプロ

名前	詳細	説明
	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。	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、「パラメータグループを変更する」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	容量に到達すると LRU 間で項目をシャッフルするバックグラウンドスレッドです。値: 0、1。
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は、2 の累乗として表されます。デフォルトは 16 (2^16)、つまり 65536 のキーです。

名前	説明
maxconns_fast	最大接続制限に達したときに新しい接続リクエストを処理する方法を変更します。このパラメータを 0 (ゼロ) に設定した場合、新しい接続がバックログキューに追加され、他の接続が終了するまで待機します。パラメータを 1 に設定した場合、ElastiCache はクライアントにエラーを送信し、すぐに接続を終了します。

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 に設定した場合、ElastiCache は低速で控えめな手法を使用して、スラブを自動的に移動します。2 に設定した場合、削除が生じると必ず ElastiCache はスラブを積極的に移動します (このモードは、テスト目的以外では推奨されません)。

名前	説明
slab_reassign	スラブの再割り当てを有効または無効にします。このパラメータを 1 に設定した場合、「slabs reassign」コマンドを使用してメモリを手動で再割り当てできます。

Memcached 1.4.5 でサポートされているパラメータ

パラメータグループファミリー: memcached1.4

Memcached 1.4.5 では、さらに次のパラメータがサポートされています。

Memcached 1.4.5 で追加されたパラメータ

名前	詳細	説明
backlog_queue_limit	デフォルト: 1024 タイプ: 整数 変更可能: いいえ	バックログキューの制限。
binding_protocol	デフォルト: auto タイプ: 文字列 変更可能: はい 変更の適用: 再起動後	バインディングプロトコル。 許可される値は ascii および auto です。 binding_protocol の値を変更する際のガイダンスについては、「パラメータグループを変更する」を参照してください。
cas_disabled	デフォルト: 0 (false) 型: ブール 変更可能: はい 変更の適用: 再起動後	1 (true) の場合、CAS (Check and Set) 操作が無効になり、格納されている項目が消費するバイト数は CAS が有効な場合より 8 バイト少なくなります。
chunk_size	デフォルト: 48 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	最も小さい項目のキー、値、およびフラグ (バイト単位) に割り当てる領域の最小量 (バイト単位)。
chunk_size_growth_factor	デフォルト: 1.25 タイプ: 浮動小数点 変更可能: はい 変更の適用: 再起動後	連続する各 memcached チャンクのサイズを制御する増加係数。各チャンクは、前のチャンクより chunk_size_growth_factor 倍大きくなります。

名前	詳細	説明
error_on_memory_exhausted	デフォルト: 0 (false) 型: ブール 変更可能: はい 変更の適用: 再起動後	1 (true) の場合、項目を保存するメモリがないと、Memcached によって項目が削除されるのではなくエラーが返されます。
large_memory_pages	デフォルト: 0 (false) 型: ブール 変更可能: いいえ	1 (true) の場合、ElastiCache は大量のメモリページを使用しようとします。
lock_down_paged_memory	デフォルト: 0 (false) 型: ブール 変更可能: いいえ	1 (true) の場合、ElastiCache はすべてのページ分割メモリをロックダウンします。
max_item_size	デフォルト: 1048576 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	クラスターに保存できる最も大きい項目のサイズ (バイト単位)。
max_simultaneous_connections	デフォルト: 65000 タイプ: 整数 変更可能: いいえ	同時接続の最大数。
maximize_core_file_limit	デフォルト: 0 (false) 型: ブール 変更可能: 変更の適用: 再起動後	1 (true) の場合、ElastiCache はコアファイルの制限を最大限に高くします。

名前	詳細	説明
memcached_connections_overhead	デフォルト: 100 タイプ: 整数 変更可能: はい 変更の適用: 再起動後	Memcached 接続および他のさまざまなオーバーヘッド用に予約されるメモリの量。このパラメータの詳細については、「Memcached 接続オーバーヘッド」を参照してください。
requests_per_event	デフォルト: 20 タイプ: 整数 変更可能: いいえ	特定の接続のイベントごとの最大リクエスト数。この制限は、リソース不足を防ぐために必要です。

Memcached 接続オーバーヘッド

各ノードで、項目の保存に使用可能なメモリは、ノード上の使用可能な合計メモリ (max_cache_memory パラメータ内) から、接続や他のオーバーヘッドに使用されているメモリ (memcached_connections_overhead パラメータ内) を引いた量です。たとえば、タイプが cache.m1.small のノードには 1300MB の max_cache_memory があるとします。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 memcached_connections_overhead の値を 13 MB に設定します。
> - 自身で作成したパラメータグループをクラスターが使用している場合、memcached_connections_overhead の値を選択した値に設定できます。

Memcached ノードタイプ固有のパラメータ

ほとんどのパラメータの値は 1 つですが、一部のパラメータには、使用されているノードタイプによって複数の値が設定されることがあります。次の表は、各ノードタイプの max_cache_memory パラメータと num_threads パラメータのデフォルト値を示しています。これらのパラメータの値は変更できません。

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.t1.micro	213	1
cache.t2.micro	555	1
cache.t2.small	1588	1
cache.t2.medium	3301	2
cache.t3.micro	512	2
cache.t3.small	1402	2
cache.t3.medium	3364	2
cache.t4g.micro	512	2
cache.t4g.small	1402	2
cache.t4g.medium	3164	2

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.m1.small	1301	1
cache.m1.medium	3350	1
cache.m1.large	7100	2
cache.m1.xlarge	14600	4
cache.m2.xlarge	33800	2
cache.m2.2xlarge	30412	4
cache.m2.4xlarge	68000	16
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.m4.large	6573	2
cache.m4.xlarge	11496	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
cache.m5.large	6537	2
cache.m5.xlarge	13248	4
cache.m5.2xlarge	26671	8
cache.m5.4xlarge	53516	16

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.m5.12xlarge	160900	48
cache.m5.24xlarge	321865	96
cache.m6g.large	6537	2
cache.m6g.xlarge	13248	4
cache.m6g.2xlarge	26671	8
cache.m6g.4xlarge	53516	16
cache.m6g.8xlarge	107000	32
cache.m6g.12xlarge	160900	48
cache.m6g.16xlarge	214577	64
cache.c1.xlarge	6600	8
cache.r3.large	13800	2
cache.r3.xlarge	29100	4
cache.r3.2xlarge	59600	8
cache.r3.4xlarge	120600	16
cache.r3.8xlarge	120600	32
cache.r4.large	12590	2
cache.r4.xlarge	25652	4
cache.r4.2xlarge	51686	8
cache.r4.4xlarge	103815	16
cache.r4.8xlarge	208144	32

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.r4.16xlarge	416776	64
cache.r5.large	13387	2
cache.r5.xlarge	26953	4
cache.r5.2xlarge	54084	8
cache.r5.4xlarge	108347	16
cache.r5.12xlarge	325400	48
cache.r5.24xlarge	650869	96
cache.r6g.large	13387	2
cache.r6g.xlarge	26953	4
cache.r6g.2xlarge	54084	8
cache.r6g.4xlarge	108347	16
cache.r6g.8xlarge	214577	32
cache.r6g.12xlarge	325400	48
cache.r6g.16xlarge	429154	64
cache.c7gn.large	3164	2
cache.c7gn.xlarge	6537	4
cache.c7gn.2xlarge	13248	8
cache.c7gn.4xlarge	26671	16
cache.c7gn.8xlarge	53516	32
cache.c7gn.12xlarge	325400	48

ノードの種類	max_cache_memory (メガバイト)	num_threads
cache.c7gn.16xlarge	108347	64

> **Note**
>
> すべての T2 インスタンスは、Amazon Virtual Private Cloud (Amazon VPC) で作成されます。

スケーリング ElastiCache (Memcached)

スケーリング ElastiCache (Memcached)

ElastiCache サーバーレスは、ワークロードのトラフィックが増減すると自動的に対応します。 ElastiCache サーバーレスキャッシュごとに、 は CPU、メモリ、ネットワークなどのリソースの使用率 ElastiCache を継続的に追跡します。これらのリソースのいずれかに制約がある場合、 ElastiCache Serverless は新しいシャードを追加し、アプリケーションをダウンタイムなく新しいシャードにデータを再分散することでスケールアウトします。キャッシュデータストレージの `BytesUsedForCache`メトリクスとコンピューティング使用量の `ElastiCacheProcessingUnits` (ECPU) をモニタリング CloudWatch することで、 のキャッシュで消費されているリソースをモニタリングできます。

スケーリング制限を設定してコストを管理する

キャッシュコストを抑えるために、キャッシュデータストレージとキャッシュの ECPU/秒の両方に使用量の上限を設定することができます。そうすることで、キャッシュ使用量が設定した上限を超えることがなくなります。

スケーリングの上限を設定した場合、キャッシュが上限に達すると、アプリケーションのキャッシュパフォーマンスが低下する可能性があります。キャッシュデータストレージの最大数を設定し、キャッシュデータストレージが最大数に達すると、 ElastiCache は LRU ロジックを使用してキャッシュ内のデータの削除を開始します。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 Memcached 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、「 ドキュメント」を参照してください。

ElastiCache Serverless による事前スケーリング

ElastiCache サーバーレスの事前スケーリング

事前スケーリングは事前ウォーミングとも呼ばれ、 ElastiCache キャッシュでサポートされる最小制限を設定できます。これらの最小値は、1 秒あたりの ElastiCache 処理単位 (ECPUsまたはデータストレージに設定できます。これは、予想されるスケーリングイベントの準備に役立ちます。例えば、

ゲーム会社が新しいゲームのローンチから最初の 1 分以内にログインが 5 倍増加すると想定している場合、この大幅な使用量の急増に対してキャッシュの準備をすることができます。

ElastiCache コンソール、CLI、または API を使用して事前スケーリングを実行できます。ElastiCache Serverless は 60 分以内にキャッシュで使用可能な ECPUs/秒を更新し、最小制限の更新が完了するとイベント通知を送信します。

事前スケーリングの仕組み

ECPU /秒またはデータストレージの最小制限がコンソール、CLI、または API を介して更新されると、その新しい制限は 1 時間以内に利用できます。 ElastiCache Serverless は空のキャッシュで 30K ECPUs/秒をサポートし、レプリカからの読み取り機能を使用すると最大 90K ECPUs /秒をサポートします。10 ~ 12 分ごとに ECPUs/秒 ElastiCache を使用できます。このスケーリング速度は、ほとんどのワークロードで十分です。今後のスケーリングイベントがこのレートを超えることが予想される場合は、ピークイベントの少なくとも 60 分前に、最小 ECPUs/秒をピーク ECPUs/秒に設定することをお勧めします。そうしないと、アプリケーションのレイテンシーが増加し、リクエストのスロットリングが発生する可能性があります。

最小制限の更新が完了すると、 ElastiCache Serverless は 1 秒あたりの新しい最小 ECPUsまたは新しい最小ストレージの計測を開始します。これは、アプリケーションがキャッシュでリクエストを実行していない場合や、データストレージの使用量が最小値を下回っている場合にも発生します。現在の設定から最小制限を引き下げると、更新は即時に行われるため、 ElastiCache サーバーレスは新しい最小制限ですぐに計測を開始します。

> **Note**
>
> - 最小使用制限を設定すると、実際の使用量が最小使用制限を下回っていても、その制限に対して課金されます。最小使用制限を超える ECPU またはデータストレージの使用量には、通常料金が課金されます。例えば、最小使用制限を 100,000 ECPUs/秒に設定すると、使用量がその最小設定よりも低い場合でも、1 時間あたり 1.224 USD 以上 (us-east-1 の ECPU 料金を使用) が課金されます。
> - ElastiCache Serverless は、キャッシュ上の集約レベルでリクエストされた最小スケールをサポートします。 ElastiCache Serverless は、スロットあたり最大 30K ECPUs/秒 (READONLY 接続を使用してレプリカから読み取るを使用する場合、90K ECPUs/秒) もサポートします。ベストプラクティスとして、アプリケーションは Redis 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. 「使用制限」で、適切なメモリまたはコンピューティング制限を設定します。
7. [プレビュー] をクリックして変更を確認し、[保存] をクリックして変更を保存します。

を使用したスケーリング制限の設定 AWS CLI

CLI を使用してスケーリング制限を変更するには、 modify-serverless-cache API を使用します。

Linux :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

Windows :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

CLI を使用してスケーリング上限を削除する

CLI を使用してスケーリング制限を削除するには、最小制限パラメータと最大制限パラメータを 0 に設定します。

Linux :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
```

```
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

Windows :

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

(ElastiCache Memcached) 独自設計型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スケーリング ElastiCache でき、使用した分に対してのみ料金が発生します。 ElastiCache を使用すると、需要に合わせてキャッシュをスケーリングできます。

> **Note**
>
> 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とき、プライマリリージョンが最初に になり、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。

以下は、実行するスケーリングアクションに適したトピックの検索に役立ちます。

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 クラスターに 60 個を超えるノードが必要な場合、または AWS リージョンに合計 300 個を超えるノードが必要な場合は、https://aws.amazon.com/contact-us/elasticache-node-limit-request/ の ElastiCache 「制限引き上げリクエスト」フォームに入力します。

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 クラスターは、1～60 個のノードを持つことができます。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。

Memcached クラスターに 60 個を超えるノードが必要な場合、または AWS リージョンに合計 300 個を超えるノードが必要な場合は、https://aws.amazon.com/contact-us/elasticache-node-limit-request/ の ElastiCache 「制限引き上げリクエスト」フォームに入力します。

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、「効率的な負荷分散のための ElastiCache クライアントの設定」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」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方向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)」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の使用」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リングするには (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、「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索 (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、「ElastiCache API の使用」を参照してください。

ElastiCache リソースのタグ付け

クラスターと他の ElastiCache リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。タグを使用すると、例えば用途別、所有者別、環境別などのさまざまな方法で AWS リソースを分類できます。これは、同じタイプのリソースが多数ある場合に役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。このトピックでは、タグとその作成方法について説明します。

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

タグの基本

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

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

タグには、ElastiCache に関連する意味はなく、完全に文字列として解釈されます。また、タグは自動的にリソースに割り当てられます。タグのキーと値は編集でき、タグはリソースからいつでも削除できます。タグの値は `null` に設定できます。特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。リソースを削除すると、リソースのタグも削除されます。さらに、レプリケーショングループでタグを追加または削除すると、そのレプリケーショングループ内のすべてのノードにもタグが追加または削除されます。

AWS Management Console、AWS CLI、および ElastiCache API を使用してタグを操作できます。

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

タグを付けることができるリソース

アカウントにすでに存在するほとんどの ElastiCache リソースにタグ付けできます。以下の表に、タグ付けをサポートするリソースを示します。AWS Management Console を使用している場合、リソースにタグを適用するには、タグエディタ を使用します。一部のリソースの画面では、リソースの作成時にリソースのタグを指定できます。たとえば、Name のキーと指定した値をタグ付けします。ほとんどの場合、リソースの作成後すぐに (リソースの作成時ではなく) コンソールによりタグが適用されます。コンソールではリソースを [Name] タグに応じて整理できますが、このタグには ElastiCache サービスに対する意味論的意味はありません。

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

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

- `CreateServerlessCache`
- `CreateCacheCluster`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `PurchaseReservedCacheNodesOffering`

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

ElastiCache リソースのタグ付けのサポート

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

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

詳細については、「リソースのタグ付けの例」を参照してください。

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

タグの制限

タグには以下のような基本制限があります。

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

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

タグ付けできる ElastiCache リソースの詳細については、「タグを付けることができるリソース」を参照してください。

リソースのタグ付けの例

- タグを使ったサーバーレスキャッシュの作成

```
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"
```

- 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 memcached \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

タグベースのアクセスコントロールポリシーの例

1. クラスターに Project=XYZ というタグがある場合にのみ、クラスターへの AddTagsToResource アクションが許可されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ"
                }
            }
        }
    ]
}
```

2. レプリケーショングループに Project タグと Service タグが含まれ、キーが Project と Service と異なる場合、レプリケーショングループからの RemoveTagsFromResource アクションが許可されます。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:RemoveTagsFromResource",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Service": "Elasticache",
                    "aws:ResourceTag/Project": "XYZ"
                },
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Project",
                        "Service"
                    ]
                }
            }
        }
    ]
}
```

3. タグが Project と Service と異なる場合にのみ、任意のリソースへの AddTagsToResource が許可されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:*:*"
            ],
            "Condition": {
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Service",
                        "Project"
                    ]
                }
```

```
            }
        }
    ]
}
```

4. リクエストタグ 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別にグループ化することで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

ン別に分類して確認できます。詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグの使用」(Use Cost Allocation Tags) を参照してください。

Memcached クラスターに ElastiCache コスト配分タグを追加できます。タグの追加やリスト、変更、削除を行った場合、そのオペレーションは、指定した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 ARN として指定されている ElastiCache リソースに適用されます。resource-type は "cluster" です。

  サンプル ARN: `arn:aws:elasticache:<region>:<customer-id>:<resource-type>:<resource-name>`

  Memcached: タグはクラスターのみに適用されます。

  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 はタグに意味論的意味を適用しません。タグは厳密に文字列として解釈されます。 AWS は 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 (Memcached) クラスターに適用されます。タグ付けされるクラスターは、ARN (Amazon リソースネーム) を使用して指定されます。

サンプル arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

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

list-tags-for-resource オペレーションを使用して、既存の ElastiCache リソースのタグを AWS CLI 一覧表示できます。

次のコードでは AWS CLI 、 を使用して、us-west-2 リージョンの Memcached クラスター`my-cluster`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
```

Windows の場合:

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

このオペレーションの出力は、リソースのすべてのタグを示した次のリストのようになります。

```
{
```

```
   "TagList": [
      {
         "Value": "10110",
         "Key": "CostCenter"
      },
      {
         "Value": "EC2",
         "Key": "Service"
      }
   ]
}
```

リソースにタグがない場合、出力は空の になります TagList。

```
{
   "TagList": []
}
```

詳細については、「」の AWS CLI 「」を参照してください ElastiCache list-tags-for-resource。

を使用したタグの追加 AWS CLI

add-tags-to-resource CLI オペレーションを使用して、既存の ElastiCache リソースにタグ AWS CLI を追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されます。

次のコードでは AWS CLI 、 を使用して、us-west-2 `my-cluster`リージョンのクラスターに、キー `Service` と を値 `elasticache`と `Region`で`us-west-2`それぞれ追加します。

Linux、macOS、Unix の場合:

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --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
```

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、「」の AWS CLI 「」を参照してください 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 (Memcached) クラスターのタグを変更できます。

タグを変更するには:

- add-tags-to-resource を使用して、新しいタグを追加するか、既存のタグに関連付けられている値を変更します。
- remove-tags-from-resource を使用して、リソースから指定したタグを削除します。

どちらのオペレーションでも、指定のクラスターのタグとその値を示すリストが出力されます。

を使用したタグの削除 AWS CLI

remove-tags-from-resource オペレーションを使用して、既存の ElastiCache (Memcached) クラスターからタグ AWS CLI を削除できます。

次のコードでは、 AWS CLI を使用して、my-clusterus-west-2 リージョンRegionからキーServiceおよび 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
```

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、「」の AWS CLI 「」を参照してください ElastiCache remove-tags-from-resource。

ElastiCache API を使用したコスト配分タグの管理

ElastiCache API を使用して、コスト配分タグを追加、変更、または削除できます。

コスト配分タグは、 ElastiCache for Memcached クラスターに適用されます。タグ付けされるクラスターは、ARN (Amazon リソースネーム) を使用して指定されます。

サンプル arn: arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster

トピック

- ElastiCache API を使用したタグの一覧表示
- ElastiCache API を使用したタグの追加
- ElastiCache API を使用したタグの変更

- ElastiCache API を使用したタグの削除

ElastiCache API を使用したタグの一覧表示

ElastiCache API を使用して、 ListTagsForResourceオペレーションを使用して既存のリソースのタグを一覧表示できます。

次のコードでは、 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>
```

ElastiCache API を使用したタグの追加

ElastiCache API を使用して、 AddTagsToResourceオペレーションを使用して既存の ElastiCache クラスターにタグを追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されます。

次のコードでは、 ElastiCache API を使用して、us-west-2 リージョンのリソースにキー Service と を値 elasticacheと Regionでus-west-2それぞれ追加my-clusterします。

```
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>
```

詳細については、「Amazon ElastiCache API リファレンスAddTagsToResource」の「」を参照してください。

ElastiCache API を使用したタグの変更

ElastiCache API を使用して、 ElastiCache クラスターのタグを変更できます。

タグの値を変更するには:

- AddTagsToResource オペレーションを使用して新しいタグと値を追加するか、既存のタグの値を変更します。
- RemoveTagsFromResource を使用して、リソースからタグを削除します。

どちらのオペレーションでも、指定のリソースのタグとその値を示すリストが出力されます。

RemoveTagsFromResource を使用して、リソースからタグを削除します。

ElastiCache API を使用したタグの削除

ElastiCache API を使用して、 RemoveTagsFromResourceオペレーションを使用して既存の ElastiCache (Memcached) クラスターからタグを削除できます。

次のコードでは、 ElastiCache API を使用して、us-west-2 `my-cluster`リージョンのクラスター`Region`からキー `Service`および 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
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Amazon ElastiCache Well-Architected レンズの使用

このセクションでは、優れたアーキテクチャの ElastiCache ワークロードを設計するための設計原則とガイダンスの集合である Amazon ElastiCache Well-Architected レンズについて説明します。

- ElastiCache レンズはAWS Well-Architected フレームワークに追加されたものです。
- 各柱には、ElastiCache アーキテクチャに関する議論を始めるのに役立つ一連の質問があります。
  - 各質問には、主なプラクティスとそのレポートスコアが記載されています。
    - 必須 - 本番前に必要 (リスクが高い場合を除く)
    - 最良 - カスタマーにとって最良の状態
    - 良い - カスタマーに推奨するもの (リスクが中程度ではない場合)
- Well-Architected の用語
  - コンポーネント — 組み合わせて要件を満たすコード、構成、AWS リソース。コンポーネントは他のコンポーネントと相互作用し、多くの場合、マイクロサービスアーキテクチャのサービスと同一視されます。
  - ワークロード 一体となって事業価値をもたらす一連のコンポーネント。ワークロードの例としては、マーケティングウェブサイト、e コマースウェブサイト、モバイルアプリのバックエンド、分析プラットフォームなどがあります。

トピック

Amazon ElastiCache Well-Architected レンズのオペレーショナルエクセレンスの柱

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 3: ElastiCache クラスターリソースを管理し、クラスター を維持する方法 up-to-date
- OE 4: ElastiCache クラスターへのクライアントの接続はどのように管理しますか？
- 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
- OE 6: 障害に対してどのように計画し、それを軽減するか。
- OE 7: Redis 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？

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 クラスターを運用する場合 ElastiCache、特定のイベントが発生したときに、オプションで通知とアラートを受け取ることができます。デフォルトでは、 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 (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
- 〔最良〕 イベントへの応答を自動化するには、 や SNS 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]: Lambda と を使用するユースケースで、 AWS Lambda、Amazon Route 53、Amazon を使用して (ElastiCache Redis OSS) (クラスターモードが無効) リードレプリカエンドポイントをモニタリングSNSしますSNS。

OE 2: 既存の ElastiCache クラスターをいつ、どのようにスケールするか。

質問レベルの紹介: ElastiCache クラスターの適切なサイズ設定は、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シング行為です。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。リソースの使用率を過小評価して修正するには、 ElastiCache を 2 つのディメンションでスケールインします。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使用は、読み取りオペレーションをレプリカノードにオフロードして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 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 Redis OSS READONLY コマンドを使用して、Redis OSSクライアントライブラリで設定できます。

[リソース]:

- 接続エンドポイントの検索
- クラスターの適切なサイズ設定
- Redis OSS READONLY コマンド

- 〔必須〕 、メモリCPU、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。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になっている場合、プライマリノードとレプリカノードは垂直方向にスケーリングできます。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
  - ElastiCache (Redis OSS) クラスターのスケーリング
  - Memcached クラスターのスケーリング ElastiCache

- [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より長い時間範囲をスキャンします。長期間の CloudWatch メトリクスを観察することで得られた学習は、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 日間利用できます。

  [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

- 〔最良〕 ElastiCache リソースが で作成されている場合は、テンプレートを使用してCloudFormation変更を実行して運用上の一貫性を維持し、管理されていない設定変更やスタックドリフトを回避するCloudFormation ことがベストプラクティスです。

  [リソース]:

  - ElastiCache の リソースタイプのリファレンス CloudFormation

- 〔最良〕 クラスターオペレーション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 と Simple Notification Service (SNS) を使用して Lambda 関数をトリガーし、 ElastiCache APIを実行してクラスターを自動的にスケーリングします。例えば、`EngineCPUUtilization` メトリクスが長期間にわたって 80% に達したときにクラスターにシャードを追加します。また、メモリベースのしきい値として `DatabaseMemoryUsedPercentages` を使用することもできます。

  [リソース]:

  - Amazon CloudWatch 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 を維持する方法 up-to-date

質問レベルの紹介: 大規模に運用する場合、すべての ElastiCacheリソースを特定して特定できることが不可欠です。新しいアプリケーション機能を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 環境タイプで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。

- 〔最良〕 利用可能な最新のエンジンバージョンで を実行し、利用可能になったらすぐにセルフサービス更新を適用します。 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 のヘルプページ
- 〔最良〕 タグを使用して ElastiCache リソースを整理します。タグは個々のノードではなくレプリケーショングループに使用します。リソースをクエリするときに表示するタグを設定したり、タグを使用して検索を実行したり、フィルターを適用できます。共通のタグセットを共有するリソースのコレクションを簡単に作成および管理するには、リソースグループを使用する必要があります。

  [リソース]:

  - タグ付けのベストプラクティス
  - 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は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、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 (応答時間など) 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、Redis OSS レプリケーションの非同期性のために、書き込み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 コマンドを活用することで、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、全体的なパフォーマンスコストで書き込みを承認するように強制できます。読み取りオペレーションにレプリカノードを使用するときは、クラスターモードが無効になっている ElastiCache リーダーエンドポイントを使用して ElastiCache (RedisOSS) クライアントライブラリで設定できます。クラスターモードが有効になっている場合は、 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はデフォルトで実装されるか、設定によって実装されます。

  [リソース]:

  - 接続エンドポイントの検索
  - READONLY

- [必須] 接続プーリングを使用します。TCP 接続を確立すると、クライアント側とサーバー側CPUの両方で時間がかかり、プーリングによって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利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使用率が低下し、クラスターで が有効になっている場合TLS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 コンソールから手動でデプロイすることも、APIs、CLI、ツールキットなどから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別に分離することも、コスト最適化の目的で組み合わせることもできます。

質問レベルのメリット: ElastiCache 環境に最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運用上の優秀性を向上させることができます。ヒューマンエラーを最小限に抑え、再現性、柔軟性、イベントへの応答時間を向上させるため、可能な限りコードとしてオペレーションを実行することをお勧めします。

ワークロードの分離要件を理解することで、ワークロードごとに専用の ElastiCache 環境を用意するか、複数のワークロードを 1 つのクラスターにまとめるか、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。自動化の考えられる手段には CloudFormation、、 AWS CLI/SDK、 などがあります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: 運用上の優秀性には、障害の潜在的な原因を特定して、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 を提供できるように、定期的な「事前償却」演習を実行して障害を予測することが含まれます。

質問レベルのメリット: 障害シナリオを事前にテストすることで、それらがワークロードにどのように影響するかを知ることができます。これにより、対応手順とその有効性を安全にテストできるだけでなく、チームはその実行に慣れておくことができます。

〔必須〕 開発/テストアカウントで定期的にフェイルオーバーテストを実行します。 TestFailover

OE 7: Redis 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、クラスターの状態とステータスを分析する機能が必要です。 ElastiCache （ Redis OSS) は、Amazon CloudWatch と Amazon Kinesis Data Firehose の両方に Redis OSS エンジンログを出力できます。

質問レベルのメリット: ElastiCache (Redis OSS) クラスターで 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するイベントに関するインサイトが得られます。Redis OSS エンジンログは、 ElastiCache イベントメカニズムでは利用できないデータを Redis OSS エンジンから直接提供します。 ElastiCache イベント (前述の OE-1 を参照) と Redis OSS エンジンログの両方を注意深く監視することで、ElastiCache サービスの観点からも Redis OSS 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 エンジンの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 と Amazon Kinesis Data Firehose のどちらが Redis OSS エンジンログの適切なターゲットであるかを判断します。
  - 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
- 〔最良〕 Amazon CloudWatch Logs を使用する場合は、Amazon CloudWatch Logs Insights を活用して Redis OSS エンジンログに重要な情報をクエリすることを検討してください。

例えば、次のような LogLevel 「」のイベントを返す Redis OSS エンジンログを含む CloudWatch ロググループに対してクエリを作成しますWARNING。

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

〔リソース]: Logs Insights を使用したログデータの分析 CloudWatch

Amazon ElastiCache Well-Architected レンズのセキュリティの柱

セキュリティの柱は、情報とシステムの保護に焦点を当てています。主なトピックは、データの機密性と完全性、権限ベースの管理による誰が何を実行できるのかの特定と管理、システムの保護、セキュリティイベントを検出するための制御の確立です。

トピック

- SEC 1: ElastiCache データへの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？
- 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？
- SEC 3: コマンドが誤って実行され、データが失われたり失敗するリスクはあるか。
- SEC 4: を使用して保管中のデータ暗号化を確保する方法 ElastiCache
- SEC 5: を使用して転送中のデータを暗号化する方法 ElastiCache
- SEC 6: コントロールプレーンリソースへのアクセスをどのように制限するか。
- SEC 7: セキュリティイベントをどのように検出して対応しているか。

SEC 1: ElastiCache データへの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、VPC、サーバーレス関数 (AWS Lambda)、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するように設計されています。最も発生しているシナリオは、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

するには、Amazon EC2 インスタンスにクラスターへのアクセスを許可する必要があります。VPC で実行されている ElastiCache クラスターにアクセスするには、クラスターにネットワーク進入を許可する必要があります。

質問レベルのメリット: クラスターへのネットワーク進入は VPC セキュリティグループによって制御されます。セキュリティグループは、Amazon EC2 インスタンスの仮想ファイアウォールとして機能し、受信トラフィックと送信トラフィックを制御します。インバウンドルールはインスタンスへの受信トラフィックを制御し、アウトバウンドルールはインスタンスからの送信トラフィックをコントロールします。の場合 ElastiCache、クラスターを起動するときは、セキュリティグループを関連付ける必要があります。これにより、インバウンドとアウトバウンドのトラフィックルールがクラスターを構成するすべてのノードに適用されるようになります。さらに、 ElastiCache は、VPC のプライベートネットワークを介して からのみ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、セキュリティグループで、送信元 IP アドレス/範囲、TCP タイプのトラフィック、および ElastiCache クラスターのポート (ElastiCache (Redis OSS) のデフォルトポート 6379) を指定するインバウンドルールを設定します。VPC 内のすべてのリソース (0.0.0.0/0) のように、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 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のインバウンドルールを定義する際には、できるだけ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 VPC ElastiCache 内の Amazon にアクセスする Lambda 関数の設定: チュートリアル: Amazon VPC ElastiCache 内の Amazon にアクセスする Lambda 関数の設定

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: ElastiCache (Redis OSS) クラスターへのアクセスを個々のクライアントレベルで制限または制御する必要がある場合は、 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、オプションでユーザーおよびユーザーグループ管理を使用して、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ます。これにより、データプレーンのセキュリティが向上します。

質問レベルのメリット: データを安全に保護するために、 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、認可されたコマンドを実行する ElastiCache 前に、クライアントが接続に使用するロールベースのアクセスコントロール (RBAC) AUTH または AUTH トークン (パスワード) の適用が含まれます。

- 〔最良] 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。RBAC を利用するには、クラスターの作成時に RBAC を選択し、転送中の暗号化を有効にする必要があります。RBAC を活用できるように、TLS をサポートする Redis OSS クライアントを使用していることを確認します。

  [リソース]:

  - (Redis OSS) のレプリケーショングループへの RBAC ElastiCache 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
- 〔最良] 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) AUTH の厳格なパスワードポリシーを維持するだけでなく、パスワード/トークンを ElastiCache いつでも最大 2 つ (2) の認証トークンに更新することをお勧めします。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 トークンの変更

SEC 3: コマンドが誤って実行され、データが失われたり失敗するリスクはあるか。

質問レベルの紹介: Redis OSS コマンドには、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に、オペレーションに悪影響を及ぼす可能性のあるものが多数あります。これらのコマ

ンドは、パフォーマンスとデータ安全性の観点から、意図しない結果をもたらす可能性があります。例えば、開発者が開発環境で日常的に FLUSHALL コマンドを呼び出している場合、間違って本番システムでこのコマンドを呼び出そうとすると、誤ってデータが失われる可能性があります。

質問レベルのメリット: ElastiCache (Redis OSS) 5.0.3 以降では、ワークロードを中断す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 OSS 5.0.3 パラメータの変更
  - Redis OSS セキュリティ

SEC 4: を使用して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) はインメモリデータストアですが、クラスターの標準オペレーションの一部として (ストレージ上で) 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、データセキュリティを強化するために、保管時の暗号化をオプションで提供します。

- 〔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) の作成時にのみ有効にできます。既存のクラスターを変更して、保管中のデータの暗号化を開始することはできません。デフォルトでは、 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]:

  - 保管時の暗号化の条件
  - 保管時の暗号化を有効にする
- [最良]メモリ内にあるデータを暗号化する Amazon EC2 インスタンスタイプ (M6g や R6g など) を活用します。可能な場合は、保管中の暗号化に独自のキーを管理することを検討してください。より厳格なデータセキュリティ環境では、 AWS Key Management Service (KMS) を使用し

てカスタマーマスターキー (CMK) を自己管理できます。と ElastiCache の統合により AWS Key Management Service、 ElastiCache (Redis OSS) クラスターの保管中のデータの暗号化に使用されるキーを作成、所有、管理できます。

[リソース]:

- 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 のコンセプト

SEC 5: を使用して転送中のデータを暗号化する方法 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ます。M6g および R6g ファミリーのインスタンスタイプには、常時オンのインメモリ暗号化も備わっています。

質問レベルのメリット: Amazon ElastiCache 転送時の暗号化は、ある場所から別の場所への転送時に、最も脆弱なポイントでデータのセキュリティを強化できるオプション機能です。

- 〔必須] 転送時の暗号化は、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どのように制限するか。

質問レベルの紹介: IAM ポリシーと ARN により、 ElastiCache (Redis OSS) のきめ細かなアクセス制御が可能になり、 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ます。

質問レベルのメリット: レプリケーショングループ、ノードなどの Amazon ElastiCache リソースの管理は、IAM ポリシーに基づいて特定のアクセス許可を持つ AWS アカウントに制約され、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 AWS を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カウントが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ているか。

質問レベルの紹介: は、RBAC ElastiCacheを有効にしてデプロイすると、 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、キーのアクセス、コマンドの実行の試みが失敗したことを特定するのに役立ちます。

さらに、 AWS 製品やサービスのリソースは、デプロイを自動化し、すべてのアクションと変更を記録して後でレビュー/監査できるようにすることで、ワークロード全体の保護に役立ちます。

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 認証と承認の失敗に関連する公開されたCloudWatch メトリクスを理解します。
  - AuthenticationFailures = Redis OSS への認証に失敗しました
  - KeyAuthorizationFailures = ユーザーがアクセス許可なしでキーにアクセスしようとして失敗した
  - 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失敗しました

  [リソース]:

  - Redis OSS のメトリクス
- [最良] これらのメトリクスにアラートと通知を設定し、必要に応じて対応することをお勧めします。

[リソース]:

- Amazon CloudWatch アラームの使用

- 〔最良] Redis OSS ACL LOG コマンドを使用して詳細を収集する

[リソース]:

- ACL ログ

- 〔最良] ElastiCache デプロイとイベントのモニターリング、ログ記録、分析に関連する AWS 製品やサービスの機能を理解します。

[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 メトリクスでの使用状況のモニタリング

Amazon ElastiCache Well-Architected レンズの信頼性の柱

トピック

REL 1: 高可用性 (HA) アーキテクチャの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。

質問レベルのメリット: 障害に対する回復力を持つようにElastiCache クラスターを設計することで、 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、主にエフェメラル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、シャードを 1 つだけ必要とする小規模なスループット要件のワークロードでも、シャードごとに最低 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. シャードされたワークロードの場合、Redis OSS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できる必要があるため、フェイルオーバーイベント中の復旧が最低 3 つのシャードで高速化されます。
  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を設定します。負荷が高い場合、 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`と `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タースケーリングをトリガーできます。
  6. 本番稼働用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: で目標復旧時点 (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の戦略に関する意思決定に役立てます。

質問レベルのメリット: インプレースRPO戦略を導入することで、ディザスタリカバリシナリオが発生した場合の事業継続性を向上させることができます。バックアップポリシーと復元ポリシーを設計すると、 ElastiCache データの目標復旧時点 (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 を理解して文書化します。
  - 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 バックアップと復元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
  - のバックアップと復元 ElastiCache (Redis OSS)

- 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
- Backup and Restore ElastiCache (Redis OSS) クラスター

REL 3: ディザスタリカバリ (DR) の要件をどのようにサポートしていますか？

質問レベルの紹介: ディザスタリカバリは、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オプションがいくつか用意されています。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 つのクロスリージョン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。 は、一部のユースケースは完全にエフェメラルで DR 戦略を必要としないのに対し、他のユースケースは逆エンドにあり、非常に堅牢な DR 戦略を必要とするという点で一意 ElastiCache 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、最低 3 つAZsが使用可能であることを確認してください。
  - 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によるマルチ AZ の有効化は 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。

質問レベルのメリット: 特定の ElastiCache (Redis OSS) クライアントライブラリと組み合わせてフェ 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実現されます。これにより、更新の頻度と方法を設定することもできます。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) 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。Redis OSSレ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ラグ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。データの安全性を高める必要があるユースケースでは、Redis OSS WAIT コマンドを使用して、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]:

  - 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 (ElastiCache Redis OSS) でのリードレプリカへの自動フェイルオーバーのテスト
  - 自動フェイルオーバーのテスト

REL 5: ElastiCache コンポーネントはスケーリング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。ElastiCache は、イン/アウト (水平) とアップ/ダウン (垂直) の 4 方向スケーリングを提供します。

質問レベルのメリット: ElastiCache デプロイのベストプラクティスに従うことで、スケーリングの柔軟性が最大になり、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
  - さらに 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、少数の大きなノードではなく、より小さなノードにバイアスを置きます。これにより、ノード障害時の影響範囲を効果的に制限できます。
- [最良] クラスターにレプリカを含めると、スケーリングイベント中の応答性が向上します

- 〔良い] クラスターモードが無効になっている場合は、リードレプリカを活用して全体的な読み取り容量を増やします。 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]:
  - ElastiCache (Redis OSS) クラスターのスケーリング
  - オンラインスケールアップ
  - Memcached ElastiCache クラスターのスケーリング

Amazon ElastiCache Well-Architected レンズのパフォーマンス効率の柱

パフォーマンス効率の柱は、IT リソースとコンピューティングリソースを効率的に使用することに重点を置いています。主なトピックには、ワークロード要件に基づいた適切なリソースの種類とサイズの選択、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 クラスターで実行速度の遅いコマンドを記録する方法
- PE8: Auto Scaling はElastiCache クラスターのパフォーマンスを向上させるのにどのように役立ちますか？

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。
- 〔最良〕 ElastiCache (Redis OSS) ワークロードでは、 などの計算コストの高いコマンドの名前を変更してKEYS、本番クラスターでブロッキングコマンドを実行するユーザーの能力を制限します。
  - 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 を使用してユーザーおよびユーザーグループを作成しCLI、そのユーザーグループを ElastiCache (RedisOSS) クラスターに関連付けることで制御できます。Redis 6 RBACでは、 OSS を有効にすると「-@dangerous」を使用できSORT、そのユーザーに対して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) ワークロードの場合は、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、`redis-cli slowlog get 10` コマンドを使用して、遅延のしきい値 (デフォルトは 10 秒) を超えた直近の 10 個のコマンドを表示できます。
  - 特定のクエリは、複雑な ElastiCache (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、クライアント数やデータ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 メトリクスでの使用状況のモニタリング
- CloudWatch メトリクスでの使用状況のモニタリング
- Amazon CloudWatch アラームの使用
- Redis 固有のパラメータ
- SLOWLOG
- Redis OSS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させます。
  - 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になっている場合、 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をバランシングす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、 は自動検出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。
- [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 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 Redis OSS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、最大メモリポリシー`redis-cli --hotkeys`が設定されている場合、 LFU 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 (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維持する必要があります。
- EElastiCache (Redis OSS) バージョン 6 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行う前に、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に ElastiCache (Redis OSS) を設定する
  - 接続エンドポイントの検索
  - 負荷分散のベストプラクティス
  - Redis 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 で一般的に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]:

- 〔必須] アプリケーションキャッシュのパフォーマンスをレイテンシーとCPU使用率の値と組み合わせて測定して収集し、 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、各データ構造の集約レイテンシーの 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) コマンドの INFO 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]:

  - 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]:

  - キャッシュ戦略
  - キャッシュのベストプラクティス
  - Amazon ElastiCache ホワイトペーパーによる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` および 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 で I/O 処理を強化してパフォーマンスを向上
  - 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量の制限に近づいている場合に予想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 (RedisOSS) 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では、有効期限 (TTL 値) が設定されたキーを削除することでメモリをvolatile-lru解放します。最も使用頻度の低い (LFU) ポリシーと最も最近使用頻度の低い (LRU) ポリシーは、使用状況に基づいてキーを削除します。
  - Memcached ワークロードの場合、各ノードのエビクションを制御するデフォルトのLRUポリシーが設定されています。Amazon ElastiCache クラスター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 ワークロードでは、 パラメータを使用してパラメータグループでDELコマンドの動作を変更できますlazyfree-lazy-user-del。
- [リソース]:
  - パラメータグループを使用したエンジンパラメータの設定
  - UNLINK
  - によるクラウド財務管理 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、使用され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。

質問レベルのメリット: 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、適切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
- 〔最良] 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大量に発生すると、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] APIsを使用して、クラスター内のデータモデルをGUI可視化します。

  [リソース]:

  - Redis OSS Commander
  - Redis OSSブラウザ
  - Redsmin

PE 7: Amazon ElastiCache クラスターで実行速度の遅いコマンドを記録する方法

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、およびエンジンの最適なパフォーマンスを妨げ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。

  - 必須パラメータは、エンジンバージョンの互換性が 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。
- 〔最良] 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、パフォーマンスの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チューニング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、コマンドの実行が遅いコマンドとして記録されるまでの Redis OSS エンジン内でのコマンドの実行時間を決定します。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 Redis OSSメモリにキャプチャされる実行速度の遅いコマンド数の上限を決定します。値が 0 の場合、キャプチャは事実上無効になります。値が大きいほど、メモリに保存されるエントリの数が増え、確認する前に重要な情報がエビクションされる可能性が低くなります。デフォルト値は 128 です。
  - デフォルト値は、ほとんどのワークロードに適しています。SLOWLOG コマンドを使用して redis-cli から拡張された時間枠内の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 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、実行速度の遅いコマンドを Redis OSSメモリに大量に保存する必要がなくなります。
- [リソース]:

- ElastiCache (Redis OSS) OSS キャッシュクラスターで Redis 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のにどのように役立ちますか？

質問レベルの紹介: Redis OSS 自動スケーリングの機能を実装することで、 ElastiCache コンポーネントを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。

質問レベルのメリット: ワークロードの急増を理解して計画することで、キャッシュのパフォーマンスとビジネス継続性を高めることができます。 ElastiCache （ Redis OSS) Auto Scaling は CPU/メモリ使用率を継続的にモニタリングして、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]:
  - 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状況のモニタリング
- [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
  - スケーラブルなターゲットの登録

Amazon ElastiCache Well-Architected レンズのコスト最適化の柱

コスト最適化の柱は、不必要なコストを回避することに焦点を当てています。主なトピックには、資金の用途の把握と管理、最適なノードタイプの選択 (ワークロードのニーズに基づくデータ階層化をサポートするインスタンスの使用)、適切な数のリソースタイプ (リードレプリカの数)、経時的な支出の分析、浪費することなくビジネスニーズを満たすためのスケーリングなどがあります。

トピック

- COST 1: リソースに関連するコストを特定して追跡する方法 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: 継続的モニタリングツールを使用して、 ElastiCache リソースに関連するコストを最適化する方法を教えてください。
- 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特定して追跡する方法 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。開発から本番稼働まで、およびリタイアまで、作成されたリソースを追跡するプロセスとツールを用意しておくと、 に関連するコストを管理するのに役立ちますElastiCache。

質問レベルのメリット: ワークロードに関連するすべてのコストを継続的に追跡するには、 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] Cloud Center of Excellence (CCoE) を設立趣意書の 1 つで作成し、組織のElastiCache 使用状況に関するメトリクスを定義、追跡、およびアクションを実行します。と 関数CCoEが存在する場合は、 に関連するコストの読み取り方法と追跡方法を知っていることを確認してください 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全体でキー ElastiCache の使用に関し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、および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非排他的な代表者が所属する部門横断的なチームで構成されています。

  [リソース]:

  - Cloud Center of Excellence の作成
  - Amazon ElastiCache 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 コストと使用状況を時間の経過とともに視覚化、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で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、請求書の費用をリソースタグ値別に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 コストを接続します。

  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 メトリクスでの使用状況のモニタリング

- 〔良い] 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 up-to-date アーキテクチャと運用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ゲートウェイ ElastiCache 、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傾向があります。
  2. クライアント、接続、セキュリティ、インメモリオペレーション、ストレージ、リソース自動化、データアクセス、管理など、ソリューションのコンポーネントをアーキテクチャ図にマッ

ピングします。各レイヤーはソリューション全体につながり、独自のニーズや機能を備えているため、全体的なコストの管理に追加したり、役立てることができます。

3. 図には、コンピューティング、ネットワーク、ストレージ、ライフサイクルポリシー、メトリクス収集の使用、アプリケーションの運用および機能 ElastiCache 要素が含まれている必要があります。
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重要です ElastiCache。

COST 2: 継続的モニタリングツールを使用して、 ElastiCache リソースに関連するコストを最適化する方法を教えてください。

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取るようにする必要があります。Amazon CloudWatch は、 ElastiCache リソースがニーズに応じ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時期を理解し、運用、可用性、レジリエンス、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 リソースを長期間識別し、運用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 ダッシュボードと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、 EngineUtilizationSwapUsage、および Evictions) は CurrConnections、スケールアップ/スケールダウン (より大きな/小さなキャッシュノードタイプを使用) またはイン/アウト (シャードの追加/レス) が必要で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 メトリクスでの使用状況のモニタリング
- モニタリングすべきメトリクス
- Amazon ElastiCache 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、バックアップは耐久性のあるストレージを提供する Amazon S3 に保存されます。障害からの復旧能力に関連するコストへの影響を理解する必要があります。
  2. 保存制限を過ぎたバックアップファイルを削除する自動バックアップを有効にします。

  [リソース]:

  - 自動バックアップのスケジュール
  - Amazon Simple Storage Service の料金表
- [最良] 十分に理解され、文書化されているワークロードのコストを管理するための計画的な戦略として、インスタンスにリザーブドノードを使用します。リザーブドノードには、ノードタイプと予約の期間 (1 年または 3 年) に応じて、前払い料金が請求されます。この料金は、オンデマンドノードで発生する 1 時間あたりの使用料金よりもはるかに安くなります。
  1. リザーブドインスタンスの要件を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は、インスタンスあたりのデータ量を大幅に増やすことができるため、サポートされるストレージデータ階層化インスタンスの数ははるかに高くなります。

- [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、アクセスのパフォーマンスコストは少なくなります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. 要件をよりよく理解するまで、非データ階層化インスタンスを使用して ElastiCache クラスターを操作する必要がある場合があります。
  2. ElastiCache クラスターのデータ使用量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

一般的なトラブルシューティング手順とベストプラクティス

トピック

- 接続の問題
- Redis 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の問題のトラブルシューティング
- 関連トピック

接続の問題

ElastiCache キャッシュに接続できない場合は、次のいずれかを検討してください。

1. TLS の使用: ElastiCache エンドポイントに接続しようとしたときに接続がハングしている場合、クライアントで TLS を使用していない可能性があります。 ElastiCache サーバーレスを使用している場合、転送中の暗号化は常に有効になります。クライアントが TLS を使用してキャッシュに接続していることを確認します。TLS 対応キャッシュへの接続の詳細については、「」を参照してください。
2. VPC: ElastiCache キャッシュには、VPC 内からのみアクセスできます。キャッシュとキャッシュにアクセスする EC2 インスタンスが同じ VPC に作成 ElastiCache されていることを確認します。または、EC2 インスタンスが存在する VPC とキャッシュを作成する VPC 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 は、セキュリティグループを使用してキャッシュへのアクセスを制御します。以下の点を考慮します。
   a. ElastiCache キャッシュで使用されるセキュリティグループが EC2 インスタンスからのインバウンドアクセスを許可していることを確認します。セキュリティグループでインバウンドルールを正しく設定する方法については、こちらを参照してください。
   b. ElastiCache キャッシュで使用されるセキュリティグループがキャッシュのポート (サーバーレスの場合は 6379 および 6380、独自設計型の場合は 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 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の詳細については、???「」を参照してください。

Redis OSS クライアントエラー

ElastiCache Serverless には、Redis OSS クラスターモードプロトコルをサポートする Redis OSS クライアントを使用してのみアクセスできます。独自設計型クラスターには、クラスター設定に応じて、Redis OSS クライアントからどちらのモードでもアクセスできます。

クライアントで Redis OSS エラーが発生した場合は、次の点を考慮してください。

1. クラスターモード: SELECT Redis OSS コマンドで CROSSLOT エラーまたはエラーが発生した場合、Redis OSS クラスタープロトコルをサポートしていない Redis OSS クライアントを使用してクラスターモードが有効なキャッシュにアクセスしようとしている可能性があります。ElastiCache Serverless は、Redis OSS クラスタープロトコルをサポートする Redis OSS クライアントのみをサポートします。「クラスターモードが無効」 (CMD) で Redis OSS 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さ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。注意点として、 ElastiCache Serverless は常にクラスターモードで動作します。複数のキーを含むマルチキーオペレーション、トランザクション、または Lua スクリプトは、関連するすべてのキーが同じハッシュスロットにある場合にのみ許可されます。

Redis 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`および CloudWatch `SuccessfulWriteRequestLatency`メトリクスを分析して、レイテンシーが ElastiCache Serverless に関連しているかどうかを確認できます。これらのメトリクスは、ElastiCache サーバーレス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いることに気付いたが、それに対応する CloudWatch `SuccessfulReadRequestLatency`およびサーバー側のレイテンシーを測定する`SuccessfulWriteRequestLatency`メトリクスの増加がない場合は、次の点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 ElastiCache していることを確認します。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ていない場合、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の詳細については、

  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の という名前が付けられます。
4. [インバウンド] タブを開き、[編集] をクリックします。
   a. [編集] を選択します。
   b. ルールの追加 を選択します。
   c. タイプ 列で、カスタムTCPルール を選択します。
   d. [ポート範囲] ボックスに、11211 と入力します。
   e. ソースボックスで、ポート範囲 (0.0.0.0/0) を持つ Anywhere 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. [ポート範囲] ボックスに、11212 と入力します。
   i. ソースボックスで、ポート範囲 (0.0.0.0/0) を持つ Anywhere を選択して、Amazon 内で起動する Amazon VPC EC2インスタンスをキャッシュ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

変動や時折のスパイクが懸念される原因になることはありません。ただし、Average統計が急激な増加を示し、持続する場合は、 AWS Health Dashboard と Personal Health Dashboard で詳細を確認する必要があります。必要に応じて、 でサポートケースを開くことを検討してください AWS Support。

レイテンシーを減らすには、次のベストプラクティスと戦略を検討してください。

- レプリカからの読み取りを有効にする: アプリケーションで許可されている場合は、Redis 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、レイテンシーを短縮することをお勧めします。有効にすると、 ElastiCache Serverless はクライアント

と同じアベイラビリティーゾーン (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結果整合性をデータに受け入れることを意味します。キーに書き込んだ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と同じ AZs にデプロイされていることを確認します。アプリケーションがキャッシュと同じ AZs にデプロイされていない場合、クライアント側のレイテンシーが高くなることがあります。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 AZs にデプロイされていることを確認します。そうしないと、キャッシュにアクセスするときにアプリケーションにクロス AZ ホップが発生し、クライアント側のレイテンシーが高くなる可能性があります。
- 接続の再利用: ElastiCache サーバーレスリクエストは、RESP プロトコルを使用した TLS 対応 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実現します。このため、接続プーリングを使用するか、既存の 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 サーバーレスは、リクエストレートの増加に応じて自動的にスケーリングします。 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突然大きく増加すると、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迅速に増加させ、リクエストレートの 2 倍になるまでに最大 10 ~ 12 分かかります。
- 長時間実行されるコマンドの検査: Lua スクリプトや大規模なデータ構造でのコマンドなど、一部の Redis OSS 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ために、 はコマンドレベルのメトリクス ElastiCache を発行します。ElastiCache 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。Serverless でリクエストがスロットリングされると、 ElastiCache Serverless `ThrottledRequests` メトリクスが増加します。 ElastiCache 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: ElastiCache (Redis OSS) では、スロットあたりのキーまたはリクエストの分散が不均等になると、ホットスロットになり、レイテンシーが増加する可能性があります。 ElastiCache Serverless は、単純な SET/GET コマンドを実行するワークロードで、1 つのスロットで最大 30,000 ECPUs/秒 (レプリカからの読み取りを使用する場合、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統一された分散を確保することをお勧めします。

ElastiCache Serverless でのスロットリングの問題のトラブルシューティング

サービス指向アーキテクチャや分散システムでは、API 呼び出しが各種サービスコンポーネントによって処理される速度を制限することをスロットリングと呼びます。これにより、スパイクが滑らかになり、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場合のより予測可能な復旧が可能になります。 ElastiCache Serverless はこれらのタイプのアーキテクチャ向けに設計されており、ほとんどの Redis OSS クライアントにはスロットリングされたリクエストの再試行が組み込まれています。ある程度のスロットリングはアプリケーションにとって必ずしも問題とはなりませんが、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。

Serverless でリクエストがスロットリングされると、 ElastiCache Serverless `ThrottledRequests` メトリクスが増加します。 ElastiCache スロットリングされたリクエストの数が多い場合は、次の点を考慮してください。

- スケーリング速度: ElastiCache サーバーレスは、より多くのデータを取り込むか、リクエストレートを上げると自動的にスケーリングします。アプリケーションが ElastiCache サーバーレスのスケーリング速度よりも速くスケールすると、リクエストがスロットリングされ、 ElastiCache サーバーレスはワークロードに合わせてスケーリングされます。 ElastiCache サーバーレスは通常、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が 2 倍になるまでに最大 10 ~ 12 分かかります。
- キーとリクエストの均一な分散: ElastiCache (Redis OSS) では、スロットあたりのキーまたはリクエストの不均等な分散により、ホットスロットが発生する可能性があります。1 つのスロットへのリクエストレートが 30,000 ECPUs/秒を超える場合、シンプルな 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がリクエストのスロットリングを引き起こす可能性があります。
- 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 Redis OSS クライアントは、読み取りをレプリカノードに転送するように「読み取りをスケーリング」するように設定できます。この機能により、読み取りトラフィックをスケーリングできます。さらに、 ElastiCache サーバーレスはレプリカリクエストからの読み取り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を有効にすると、

シンプルな SET/GET コマンドを使用するワークロードに対して、1 つのスロットで最大 90,000 ECPUs/秒を実現できます。

関連トピック

- その他のトラブルシューティング手順
- the section called “ベストプラクティスとキャッシュ戦略”

その他のトラブルシューティング手順

の永続的な接続の問題をトラブルシューティングする際は、以下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 ACL
- ルートテーブル
- DNS 解決
- サーバー側の診断に関する問題の特定
- ネットワーク接続性の検証
- ネットワーク関連の制限
- CPU 使用率
- サーバー側からの接続が終了している
- Amazon EC2 インスタンスのクライアント側のトラブルシューティング
- 1 つのリクエストを完了するのにかかった時間の解読

セキュリティグループ

セキュリティグループは、 ElastiCache クライアント (EC2 インスタンス、 AWS Lambda 関数、Amazon ECS コンテナなど) 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 側で制限に達したかどうかを確認する方法については、トラブルシューティングセクションを参照してください。

クライアントと ElastiCache クラスターに同時に 1 つのセキュリティグループを割り当てることも、それぞれに個別のセキュリティグループを割り当てることもできます。

どちらの場合も、送信元からの ElastiCache ポート上の TCP アウトバウンドトラフィックと、同じポート上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、Redis OSS の場合は 6379 です。デフォルトで、セキュリティグループはすべてのアウトバウンドトラフィックを許可します。この場合、ターゲットセキュリティグループのインバウンドルールのみが必要です。

詳細については、「Amazon VPC 内の ElastiCache クラスターにアクセスするためのアクセスパターン」を参照してください。

ネットワーク ACL

ネットワークアクセスコントロールリスト (ACL) は、ステートレスルールです。トラフィックを成功させるには、両方向 (インバウンドとアウトバウンド) で許可する必要があります。ネットワーク ACL は、特定のリソースではなく、サブネットに割り当てられます。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 ACL はすべてのトラフィックを許可します。ただし、トラフィックを拒否または許可するようにカスタマイズすることは可能です。さらに、ACL ルールの評価はシーケンシャルです。つまり、トラフィックに一致する最も小さい番号を持つルールで、それが許可または拒否されます。Redis 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]。
- [Protocol]: 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のルールを作成する)
- [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]。
- [Protocol]: 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 IPs 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可能性があることに注意してください)。
- [許可/拒否]: 許可

ElastiCache ネットワーク ACL:

- インバウンドルール:
- ルール番号: 好ましくは拒否ルールよりも低い。
- [タイプ]: [カスタム TCP ルール]。
- [Protocol]: TCP
- [Port Range]: 6379
- ソース: 0.0.0.0/0 (または ElastiCache クラスターサブネットの個別のルールを作成する)
- [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]。
- [Protocol]: 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 IPs 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可能性があることに注意してください)。
- [許可/拒否]: 許可

詳細については、「ネットワーク ACL」を参照してください。

ルートテーブル

ネットワーク ACL と同様に、各サブネットは異なるルートテーブルを持つことができます。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で相互に到達できることを確認してください。

複数の VPC、動的ルーティング、またはネットワークファイアウォールを含む、より複雑な環境は、トラブルシューティングが困難になることがあります。「ネットワーク接続性の検証」を参照して、ネットワーク設定が適切であることを確認してください。

DNS 解決

ElastiCache は、DNS 名に基づいてサービスエンドポイントを提供します。使用可能なエンドポイントは、`Configuration`、`Primary`、`Reader`、および `Node` エンドポイントです。詳細情報については、「接続エンドポイントの検索」を参照してください。

フェイルオーバーまたはクラスターの変更の場合、エンドポイント名に関連付けられたアドレスが変更され、自動的に更新されます。

カスタム DNS 設定 (VPC DNS サービスを使用しない) では、 ElastiCacheが提供する DNS 名を認識しない場合があります。`dig` (次に示すように) 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 サービスを介して名前解決を強制することもできます。

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com @169.254.169.253
example-001.tihewd.0001.use1.cache.amazonaws.com.
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、接続の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 使用率: 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、Redis OSS プロセス専用の CPU 使用率と、すべての vCPU のCPUUtilization使用率を提供します。vCPUs 複数の vCPU を持つノードは、通常、CPUUtilization と EngineCPUUtilization で異なる値を持ち、ここで 2 番目が一般的に高くなります。高い EngineCPUUtilization は、リクエスト数の上昇や、完了までに多くの CPU 時間を要する複雑なオペレーションが原因である可能性があります。両方を特定するには、以下を使用します。
  - リクエスト数の上昇: EngineCPUUtilization パターンに一致する他のメトリクスでの上昇がないか確認します。役に立つメトリクスは次のとおりです。
    - CacheHits と CacheMisses: 成功したリクエスト数、またはキャッシュで有効な項目を見つけ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で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、レプリカノードを自動的に検出し、READONLY 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、多くの CPU オーバーヘッドの発生を意味します。さらに、新しい接続の作成に必要な TCP スリーウェイハンドシェイクは、全体的な応答時間に悪影響を及ぼします。

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。Redis OSS は、少数の currConnections で最適なパフォーマンスを提供します。CurrConnection を数十または数百のオーダーで維持すると、クライアントバッファのような個別の接続や、接続に役立つ CPU サイクルをサポートするためのリソースの使用量が最小限になります。

- ネットワークスループット:
  - 帯域幅を決定します。 ElastiCache ノードのネットワーク帯域幅は、ノードサイズに比例します。アプリケーションの特性が異なるため、結果はワークロードに応じて異なる場合があります。たとえば、小さなリクエストの割合が高いアプリケーションは、ネットワークのスループットよりも CPU 使用率に影響を及ぼす傾向がありますが、キーが大きいほどネットワーク使用率が高くなります。そのため、制限をよりよく理解するために、実際のワークロードでノードをテストすることをお勧めします。

    アプリケーションからの負荷をシミュレートすると、より正確な結果が得られます。ただし、ベンチマークツールは、制限についての良いアイデアを提供することができます。
  - リクエストが主に読み取りの場合、読み取りオペレーションでレプリカを使用すると、プライマリノードの負荷が軽減されます。ユースケースが主に書き込みの場合、多くのレプリカを使用するとネットワークの使用が増大します。プライマリノードに書き込まれるすべてのバイトについて、N バイトがレプリカに送信され、ここで N はレプリカの数になります。書き込み集約型ワークロードのベストプラクティスは、 ElastiCache (Redis OSS) をクラスターモードを有効にして使用して、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の量をそれぞれ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 コマンドは単一のスレッドで提供されます。つまり、リクエストは順番に処理されます。単一のスローコマンドは、他のリクエストや接続に影響を及ぼし、タイムアウトが発生する可能性があります。複数の値、キー、またはデータ型に作用するコマンドの使

用は、慎重に行う必要があります。接続は、パラメータの数や入出力値のサイズに応じて、ブロックまたは終了できます。

評判の悪い例は、KEYS コマンドです。これは、指定されたパターンを検索するキースペース全体をスイープし、その実行中に他のコマンドの実行をブロックします。Redis OSS は「Big 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。すべてのコマンドのリストとそれぞれの時間の複雑さを確認するには、https://redis.io/commands にアクセスしてください。

潜在的な問題の例:

- Lua スクリプト: Redis OSS には Lua インタープリタが埋め込まれており、サーバー側でスクリプトを実行できます。Redis OSS の Lua スクリプトはエンジンレベルで実行され、定義上アトミックです。つまり、スクリプトの実行中に他のコマンドやスクリプト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 Redis OSS エンジン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 秒に制限されています。キースペースに書き込まれていないスクリプトは、5 秒後に自動的に終了します。データの破損や不整合を避けるため、スクリプトの実行が 5 秒以内に完了せず、実行中に書き込みがあった場合、ノードはフェイルオーバーします。トランザクションは、Redis OSS で複数の関連するキー変更の一貫性を保証するための代替手段です。トランザクションは、コマンドのブロックの実行を可能にし、既存のキーの変更を監視します。トランザクションの完了前に監視キーのいずれかが変更された場合、すべての変更は破棄されます。
- アイテムの一括削除: DEL コマンドは、削除するキー名である複数のパラメータを受け入れます。削除オペレーションは同期的であり、パラメータのリストが大きい場合、または大きなリスト、セット、ソートされたセット、またはハッシュ (いくつかのサブ項目を保持するデータ構造) が含まれている場合、多くの CPU 時間がかかります。言い換えれば、単一のキーを削除しても、多くの要素がある場合、多くの時間がかかることがあります。の代替DEL手段は です。これはUNLINK、Redis OSS 4 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される必要があります。 ElastiCache (Redis OSS) 6x 以降、lazyfree-lazy-user-delパラメータは、有効になっているUNLINKときにDELコマンドをのように動作させます。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、同様に機能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 使用率に影響します。

  CPU 使用率だけが接続の問題の原因ではありませんが、複数のキーで単一または少数のコマンドを処理するのに時間がかかりすぎると、他のリクエストが失敗し、CPU 使用率が増加する可能性があります。

キーの数とそのサイズは、コマンドの複雑さ、また結果として完了時間に影響します。

複数のキーに対して作用できるコマンドのその他の例:
HMGET、HMSET、MSETNX、PFCOUNT、PFMERGE、SDIFF、SDIFFSTORE、SINTER、SINTERSTORE
または ZINTERSTORE。

- 複数のデータ型で動作するコマンド: Redis OSS 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。
- 他の大きなデータ構造: ハッシュに加えて、他のデータ構造では CPU 集中的になる可能性があります。セット、リスト、ソートされたセット、およびハイパーログログも、そのサイズと使用されるコマンドによっては、操作に多くの時間がかかることがあります。これらのコマンドの詳細については、「https://redis.io/commands」を参照してください。

ネットワーク接続性の検証

DNS 解決、セキュリティグループ、ネットワーク ACL、およびルートテーブルに関連するネットワーク設定を確認した後、VPC Reachability Analyzer とシステムツールを使用して接続性を検証できます。

Reachability Analyzer は、ネットワーク接続性をテストし、すべての要件と許可が満たされているかどうかを確認します。以下のテストでは、VPC で使用可能な ElastiCache ノードの 1 つの ENI ID (Elastic Network Interface Identification) が必要になります。それを見つけるには、以下の操作を行います。

1. https://console.aws.amazon.com/ec2/v2/home?#NIC: にアクセスします。
2. ElastiCache クラスター名または以前に DNS 検証から取得した IP アドレスでインターフェイスリストをフィルタリングします。
3. ENI IDを書き留めるか、保存してください。複数のインターフェイスが表示されている場合は、説明を確認して正しい ElastiCache クラスターに属していることを確認し、そのうちの 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/home?#ReachabilityAnalyzer で分析パスを作成し、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 インスタンスまたはネットワークインターフェイスで実行され、awsvpc ネットワークを使用する AWS Fargate Amazon ECS などの別のサービスを使用している場合はインスタンスを選択します。また AWS Lambda、それぞれのリソース ID (EC2 インスタンスまたは ENI ID) を選択します。
- [送信先タイプ]: [ネットワークインターフェイス] を選択し、リストから [Elistiache 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。
- [Protocol]: TCP

分析パスを作成し、結果まで数分待ちます。ステータスが到達不能の場合は、解析の詳細を開き、[解析エクスプローラ] で、リクエストがブロックされた場所の詳細を確認してください。

到達可能性テストに合格した場合は、OS レベルでの検証に進みます。

ElastiCache サービスポートでの TCP 接続を検証するには: Amazon Linux では、 Npingが パッケージで利用nmapでき、 ElastiCache ポートでの TCP 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ラウンドトリップ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。

nping でのテストが失敗し、[VPC Reachability Analyzer] テストに合格した場合は、オペレーティングシステムレベルで考えられる、ホストベースのファイアウォールルール、非対称ルーティングルール、またはその他の制限を確認するよう、システム管理者に依頼してください。

ElastiCache コンソールで、 ElastiCache クラスターの詳細で転送中の暗号化が有効になっているかどうかを確認します。転送中の暗号化が有効になっている場合は、次のコマンドを使用して TLS セッションを確立できるかどうかを確認します。

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

接続と TLS ネゴシエーションが成功すると、広範な出力が期待されます。最後の行で利用可能な戻りコードを確認してください。値は `0`（ok）である必要があります。openssl が何か異なるものを返す場合は、https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS でエラーの理由を確認します。

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、 ElastiCache クラスターモードが有効になっています。
- アプリケーションが TLS/SSL をサポートしておらず、転送 ElastiCache 中の暗号化が有効になっています。
- アプリケーションは TLS/SSL をサポートしていますが、正しい設定フラグまたは信頼できる証明権限がありません。

ネットワーク関連の制限

- 最大接続数: 同時接続にはハード制限があります。各 ElastiCache ノードでは、すべてのクライアントで最大 65,000 の同時接続が可能です。この制限は、 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、同じ送信元から同じ送信先 IP (ElastiCache ノード) とポートへの 28231 接続が許可されます。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 OSS の次のメトリクスをチェックして、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、複数のレプリカ、および集中的な書き込みリクエストがあるクラスターは、レプリケーションのバックログに対処できない場合があります。このような場合は、スケールアップ (ノードタイプを変更する)、スケールアウト (クラスターモードが有効なクラスターにシャードを追加する)、レプリカの数を減らす、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。

  プラトーが長期間にわたって観察される場合、レプリケーションの遅れ、キャッシュで使用されるバイト数の増加、解放可能なメモリの低下、高いスワップおよび CPU 使用率がそれに続きます。また、Amazon EC2 インスタンスには、[ENA ドライバーメトリクス] を介して追跡できるネットワーク制限があります。拡張ネットワーキングサポートおよび ENA ドライバー 2.2.10 以降を備えた Linux インスタンスは、次のコマンドを使用して制限カウンターを確認できます。

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU 使用率

CPU 使用率メトリクスは調査の出発点であり、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 (UTC) に起こり、完了までに 4.8秒 (4.823ms) かかり、クライアント 1.2.3.4 からリクエストされた KEYS コマンドによって発生しました。

Linux では、タイムスタンプはコマンド date で変換できます。

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

Python の場合:

```
>>> from datetime import datetime
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

または、 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) の時間の複雑さがあるコマンドが頻繁に報告される場合は、前述の CPU 使用率が高いかどうかについて他の要因を確認してください。

- レイテンシーメトリクス: ElastiCache (Redis 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、`eval`、 に関連する`evalsha`。
  - GeoSpatialBasedCmdsLatency: `geodist`, `geohash`, `geopos`, `georadius`, `georadiusbymember`, `geoadd`;
  - GetTypeCmdsLatency: データ型に関係なくコマンドを読み取る。

- HashBasedCmdsLatency: hexists, hget, hgetall, hkeys, hlen, hmget, hvals, hstrlen, hdel, hincrby, hincrbyfloat, hmset, hset, hsetnx;
- HyperLogLogBasedCmdsLatency: pfselftest, pfcount, pfdebug, pfadd, pfmerge;
- KeyBasedCmdsLatency: さまざまなデータ型に対して動作できるコマンド: dump、exists、keysobject、、pttl、randomkey、ttltypedel、expire、、expireat、mov 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。
- バックアップとレプリケーション: ElastiCache (Redis OSS) 2.8.22 より前のバージョンでは、フォークプロセスを使用してバックアップを作成し、レプリカとの完全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 OSS 2.8.22 以降、 はフォークレスバックアップとレプリケーションの方法 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のタイムアウトが発生する可能性があります。バックアップウィンドウの間にクライアントの障害が発生したか、または SaveInProgress メトリクスが期間内で 1 であったかどうかを常に確認してください。クライアントの問題やバックアップ障害の可能性を最小限にするために、使用率の低い期間でバックアップウィンドウをスケジュールすることをお勧めします。

サーバー側からの接続が終了している

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「timeout」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内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、メモリを解放して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: 1 つの入力リクエストの最大サイズ。

- client-output-buffer-limit-normal-soft-limit: クライアント接続のソフト制限。で定義された秒単位の時間を超えてソフトリミットを超えたままの場合、normal-soft-seconds またはハードリミットに達した場合 client-output-buffer-limit、接続は終了します。
- client-output-buffer-limit-normal-soft-seconds: client-output-buffer-limit-normal-soft-limit; を超える接続に許容される時間
- client-output-buffer-limit-normal-hard-limit: この制限に達した接続は直ちに終了します。

通常のクライアントバッファに加えて、次のオプションは、レプリカノードと Pub/Sub (Publish/Subscribe) クライアントのバッファを制御します。

- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-replica-soft-seconds;
- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-pubsub-soft-limit;
- client-output-buffer-limit-pubsub-soft-seconds;
- client-output-buffer-limit-pubsub-hard-limit;

Amazon EC2 インスタンスのクライアント側のトラブルシューティング

クライアント側の負荷と応答性は、 へのリクエストにも影響します ElastiCache。断続的な接続性またはタイムアウトの問題のトラブルシューティングを行う際には、EC2 インスタンスおよびオペレーティングシステムの制限を慎重に確認する必要があります。観察すべきいくつかの重要なポイント：

- CPU:
  - EC2 インスタンスの CPU 使用率: CPU が飽和していない、または 100% 近くではないことを確認します。履歴分析は を介して行うことができますが CloudWatch、データポイントの詳細度は 1 分 (詳細モニタリングが有効) または 5 分であることに注意してください。
  - [バースト可能な EC2 インスタンス] を使用する場合は、CPU クレジット残高が枯渇していないことを確認してください。この情報は、 `CPUCreditBalance` CloudWatch メトリクスで確認できます。
  - 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 のインスタンスタイプ」を参照してください。
- ena 拡張ネットワークドライバーのインスタンスで、タイムアウトまたは超えられた制限がないか [ENA 統計] を確認してください。次の統計情報は、ネットワーク制限の飽和状態を確認するのに役立ちます。
  - bw_in_allowance_exceeded/bw_out_allowance_exceeded: 過剰なインバウンドまたはアウトバウンドのスループットのためにシェーピングされたパケット数;
  - conntrack_allowance_exceeded: セキュリティグループの [接続追跡制限] のためにドロップされたパケット数。この制限が飽和すると、新しい接続は失敗します。
  - linklocal_allowance_exceeded: インスタンスメタデータ、VPC DNS 経由の NTP への過剰なリクエストによりドロップされたパケット数。制限は、すべてのサービスで毎秒 1024 パケットです。
  - pps_allowance_exceeded: 1 秒あたりの過剰なパケット比率のためにドロップされたパケット数。PPS 制限は、ネットワークトラフィックが 1 秒あたり数千または数百万の非常に小さなリクエストで構成されている場合にヒットする可能性があります。 ElastiCache トラフィックは、 MGETではなく、 のように複数のオペレーションを一度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有効に使用するために最適化できます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 エンジンに到達してリターンを得るのにかかった時間を理解するのに役立つツールです。次の例では、次のコマンドで作成された 1 つのリクエストを強調表示します。

```
$ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
+PONG
```

上記のコマンドと並行して、tcpdump が実行中であり、次のように返されました。

```
$ sudo tcpdump -i any -nn port 6379 -tt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
1609428918.917869 IP 172.31.11.142.40966
```

```
    > 172.31.11.247.6379: Flags [S], seq 177032944, win 26883, options [mss
 8961,sackOK,TS val 27819440 ecr 0,nop,wscale 7], length 0
1609428918.918071 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [S.], seq
 53962565, ack 177032945, win
    28960, options [mss 1460,sackOK,TS val 3788576332 ecr 27819440,nop,wscale 7],
 length 0
1609428918.918091 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 1, win
 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918122
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [P.], seq 1:6, ack 1, win 211,
 options [nop,nop,TS val 27819440 ecr 3788576332], length 5: RESP "ping"
1609428918.918132 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [F.], seq 6, ack
    1, win 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918240 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [.], ack 6, win
 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918295
    IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [P.], seq 1:8, ack 7, win 227,
 options [nop,nop,TS val 3788576332 ecr 27819440], length 7: RESP "PONG"
1609428918.918300 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 8, win
    211, options [nop,nop,TS val 27819441 ecr 3788576332], length 0
1609428918.918302 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [F.], seq 8, ack
 7, win 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918307
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 9, win 211, options
 [nop,nop,TS val 27819441 ecr 3788576332], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

上記の出力から、TCP スリーウェイハンドシェイクが 222 マイクロ秒 (918091 ~ 917869) で完了し、ping コマンドが送信され、173 マイクロ秒 (918295 ~ 918122) で返されたことを確認できます。

リクエストから接続を閉じるまで、438 マイクロ秒 (918307 ~ 917869) かかりました。これらの結果では、ネットワークとエンジンの応答時間が良好であることを確認し、調査は他のコンポーネントに焦点を当てることができます。

- オペレーティングシステム上: Strace は、OS レベルでのタイムギャップを特定するのに役立ちます。実際のアプリケーションの分析では、より広範で特殊なアプリケーションプロファイラやデバッガを使用することをお勧めします。次の例は、基本オペレーティングシステムコンポーネントが予期したとおりに動作しているかどうかを示しています。そうでない場合、さらに調査が必要に

なることがあります。で同じ Redis OSS PING コマンドを使用すると、strace次のことが可能になります。

```
$ 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 のインスタンス化と名前解決 (697712 から 717890 まで) には多くの時間 (約 20ms) がかかりました。その後、TCP ソケットの作成には 2ms (745659 から 747858)、リクエストに対する応答の送信と受信には 0.4ms (747858 から 748330) が必要でした。

Amazon ElastiCache でのセキュリティ

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS と顧客の間の責任共有です。責任共有モデルでは、この責任が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されています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティー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 ElastiCache に適用されるコンプライアンスプログラムについては、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ - ユーザーの責任は、使用する AWS サービスに応じて異なります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon ElastiCache 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックで、セキュリティおよびコンプライアンスの目的を満たすように、Amazon ElastiCache を設定する方法について説明します。Amazon ElastiCache リソースのモニタリングやセキュリティ確保に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- Amazon ElastiCache でのデータ保護
- インターネットトラフィックのプライバシー
- Amazon の Identity and Access Management ElastiCache
- Amazon のコンプライアンス検証 ElastiCache
- Amazon ElastiCache s の耐障害性
- AWS ElastiCache でのインフラストラクチャセキュリティ
- でのサービスの更新 ElastiCache

Amazon ElastiCache でのデータ保護

AWS 責任共有モデルは、AWS ElastiCache (ElastiCache) のデータ保護に適用されます。このモデルで説明したように、AWS は、すべての AWS クラウドを実行するグローバルインフラストラクチャを保護する責任を負います。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用する AWS のサービスに対するセキュリティの設定と管理タスクが含まれます。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。

データ保護の目的で、AWS アカウントの認証情報を保護し、個々のアカウントを AWS Identity and Access Management (IAM) で設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- TLS を使用して AWS リソースと通信します。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS 暗号化ソリューションを AWS のサービス内のすべてのデフォルトのセキュリティ管理と一緒に使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これにより、Amazon S3 に保存される個人データの検出と保護が支援されます。

顧客のアカウント番号などの機密の識別情報は、[名前] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK を使用して ElastiCache や他の AWS のサービスを使用する場合も同様です。ElastiCache や他のサービスに入力したすべてのデータは、診断ログに取り込まれる可能性があります。外部サーバーへの URL を指定するときは、そのサーバーへのリクエストを検証するための認証情報を URL に含めないでください。

トピック

- Amazon のデータセキュリティ ElastiCache

Amazon のデータセキュリティ ElastiCache

データの安全性を維持するために、Amazon ElastiCache と Amazon EC2 はサーバー上のデータの不正アクセスから保護するメカニズムを提供します。

Amazon ElastiCache (Memcached) には、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も用意されています。

- 転送時の暗号化では、ある場所から別の場所に移動するデータ (クラスターのノード間、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。

トピック

- ElastiCache 転送時の暗号化 (TLS)
- での保管時の暗号化 ElastiCache

ElastiCache 転送時の暗号化 (TLS)

データを安全に保つために、Amazon ElastiCache と Amazon EC2はサーバー上のデータの不正アクセスから保護するメカニズムを提供します。転送時の暗号化機能を提供することで、 は、ある場所から別の場所に移動しているデータの保護に役立つツール ElastiCache を提供します。

すべてのサーバーレスキャッシュで、転送時の暗号化が有効になっています。独自設計型クラスターの場合、 `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時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
- 転送時の暗号化の条件
- 転送時の暗号化のベストプラクティス
- 転送時の暗号化を有効にする
- Openssl を使用して送信中の暗号化を有効にしたノードへの接続
- 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 転送時の暗号化は、ある場所から別の場所への転送中に、最も脆弱なポイントでデータのセキュリティを強化できる機能です。エンドポイントでデータの暗号化と復号を行うにはある程度の処理が必要であるため、転送時の暗号化を有効にするとパフォーマンスに影響を及ぼす可能性があります。転送時の暗号化の使用時と未使用時でデータのベンチマークを取得して、ユースケースにおけるパフォーマンス影響を判断する必要があります。

ElastiCache 転送時の暗号化には、次の機能が実装されています。

- 暗号化されたクライアント接続 — キャッシュノードへのクライアント接続はTLS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。

転送時の暗号化の条件

独自設計型クラスターの実装を計画するときは、Amazon の転送 ElastiCache 時の暗号化に関する以下の制約事項に留意する必要があります。

- 転送時の暗号化は、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。
- 転送時の暗号化は、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。
- 転送時の暗号化は、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。
- 転送時の暗号化は、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接続を永続化することで、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。

転送時の暗号化を有効にする

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。

step-by-step プロセスについては、「Memcached クラスターの作成 (コンソール)」を参照してください。

Openssl を使用して送信中の暗号化を有効にしたノードへの接続

転送時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 () と連携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 で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 で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/mem-ug/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。

での保管時の暗号化 ElastiCache

データを安全に保護するために、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 VPC と ElastiCache のセキュリティ」および「Amazon の Identity and Access Management ElastiCache」を参照してください。

- 同期およびスワップオペレーション中のディスク

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、 Key Management Service (AWS KMS) で独自の対称カスタマーマネージド KMS キーを使用する機能を提供します。AWS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。

保管時の暗号化は、キャッシュに対してその作成時にのみ有効にできます。データの暗号化と復号を行うにはある程度の処理が必要であるため、保管時の暗号化を有効にすると、これらのオペレーションの実行中のパフォーマンスに影響を与える可能性があります。保管時の暗号化の使用時と未使用時でデータのベンチマークを取得して、ユースケースにおけるパフォーマンスの影響を判断する必要があります。

トピック

- 保管時の暗号化の条件
- AWS KMS からのカスタマーマネージドキーの使用
- 保管時の暗号化を有効にする
- 以下の資料も参照してください。

保管時の暗号化の条件

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事項に留意する必要があります。

- 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。
- 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。

AWS KMS からのカスタマーマネージドキーの使用

ElastiCache は、保管時の暗号化用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 Key Management Service デベロッパーガイドの「AWS KMS キー」を参照してください。キーは、 で使用する前に AWS KMS で作成する必要があります ElastiCache。

AWS KMS ルートキーの作成方法については、「 Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。 AWS

ElastiCache では、 を AWS KMS と統合できます。詳細については、AWS Key Management Service デベロッパーガイドの「付与の使用」を参照してください。Amazon と AWS KMS 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。

`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、条件キーの値に両方の ViaService 名前を含めます: `elasticache.AWS_region.amazonaws.com`および `dax.AWS_region.amazonaws.com`。詳細については、「kms:ViaService」を参照してください。

を使用してAWS CloudTrail、Amazon がユーザーに代わって ElastiCache に送信する AWS Key Management Service 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 コールには、対応する CloudTrail ログがあります。KMS API コールを呼び出すことで、 ListGrants が ElastiCache 作成する権限を確認することもできます。

- 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旧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は AWS KMS ルートキーのプロパティを保持するため、ローテーションは ElastiCache データへのアクセス能力には影響しません。暗号化された Amazon ElastiCache キャッシュは、新しいルートキーの作成と古いキーへの参照の更新を含む手動キーローテーション

をサポートしていません。詳細については、「 Key Management Service デベロッパーガイド」の AWS 「KMS キーのローテーション」を参照してください。 AWS

- 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。
- AWS KMS の許可と制限の詳細については、AWS 「 Key Management Service デベロッパーガイド」の「制限」を参照してください。

保管時の暗号化を有効にする

すべてのサーバーレスキャッシュでは、保管時の暗号化が有効になっています。

ElastiCache キャッシュを作成するときに、保管時の暗号化を有効にできます。これを行うには AWS Management Console、、 AWS CLI、または ElastiCache API を使用します。

キャッシュを作成するときに、以下のオプションのいずれかを選択できます。

- デフォルト - このオプションでは、サービス管理の保存時の暗号化が使用されます。
- カスタマーマネージドキー – このオプションを使用すると、 AWS KMS からキー ID/ARN を指定して保管時の暗号化を行うことができます。

AWS KMS ルートキーの作成方法については、「 Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。 AWS

目次

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所有の 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。

以下の資料も参照してください。

- Amazon VPC と ElastiCache のセキュリティ
- Amazon の Identity and Access Management ElastiCache

インターネットトラフィックのプライバシー

Amazon ElastiCache では、以下の方法によりキャッシュデータを不正なアクセスからセキュリティで保護します。

- Amazon VPC と ElastiCache のセキュリティ では、インストールに必要なセキュリティグループのタイプを説明します。
- Amazon の Identity and Access Management ElastiCache は、ユーザー、グループ、グループ、ロールの付与と制限のためのものです。

Amazon VPC と ElastiCache のセキュリティ

データのセキュリティは重要であるため、ElastiCache には、データへのアクセス権限を持つユーザーを制御するための手段が用意されています。データへのアクセスを制御する方法は、Amazon Virtual Private Cloud (Amazon VPC) または Amazon EC2-Classic でクラスターを起動したかどうかによって決まります。

> **Important**
>
> ElastiCache クラスターの起動のために Amazon EC2-Classic の使用を非推奨にしました。現行世代のすべてのノードは Amazon Virtual Private Cloud のみで起動されます。

Amazon Virtual Private Cloud (Amazon VPC) サービスは、従来のデータセンターに非常によく似た仮想ネットワークを定義します。お客様が Amazon VPC を設定すると、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能になります。仮想ネットワークにキャッシュクラスターを追加でき、Amazon VPC のセキュリティグループを使用して、キャッシュクラスターへのアクセスを制御できます。

このセクションでは、Amazon VPC 内で手動で ElastiCache クラスターを設定する方法を説明します。この情報は、ElastiCache と Amazon VPC との連携について理解を深めたいユーザーを対象としています。

トピック

- ElastiCache と Amazon VPC について理解する
- Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン
- Virtual Private Cloud (VPC) の作成
- Amazon VPC で実行されるキャッシュへの接続

ElastiCache と Amazon VPC について理解する

ElastiCache は Amazon Virtual Private Cloud (Amazon VPC) と完全に統合されています。ElastiCache ユーザーにとって、これは次のことを意味します。

- AWS アカウントが EC2-VPC プラットフォームのみをサポートしている場合、ElastiCache は常に Amazon VPC でクラスターを起動します。
- AWS を初めて利用する場合、クラスターは Amazon VPC にデプロイされます。デフォルト VPC が自動的に作成されます。
- デフォルト VPC をお持ちのお客様が、クラスター起動時にサブネットを指定しなかった場合は、そのクラスターはお客様のデフォルト Amazon VPC で起動されます。

詳細については、「サポートされているプラットフォームとデフォルト VPC があるかどうかを確認する」を参照してください。

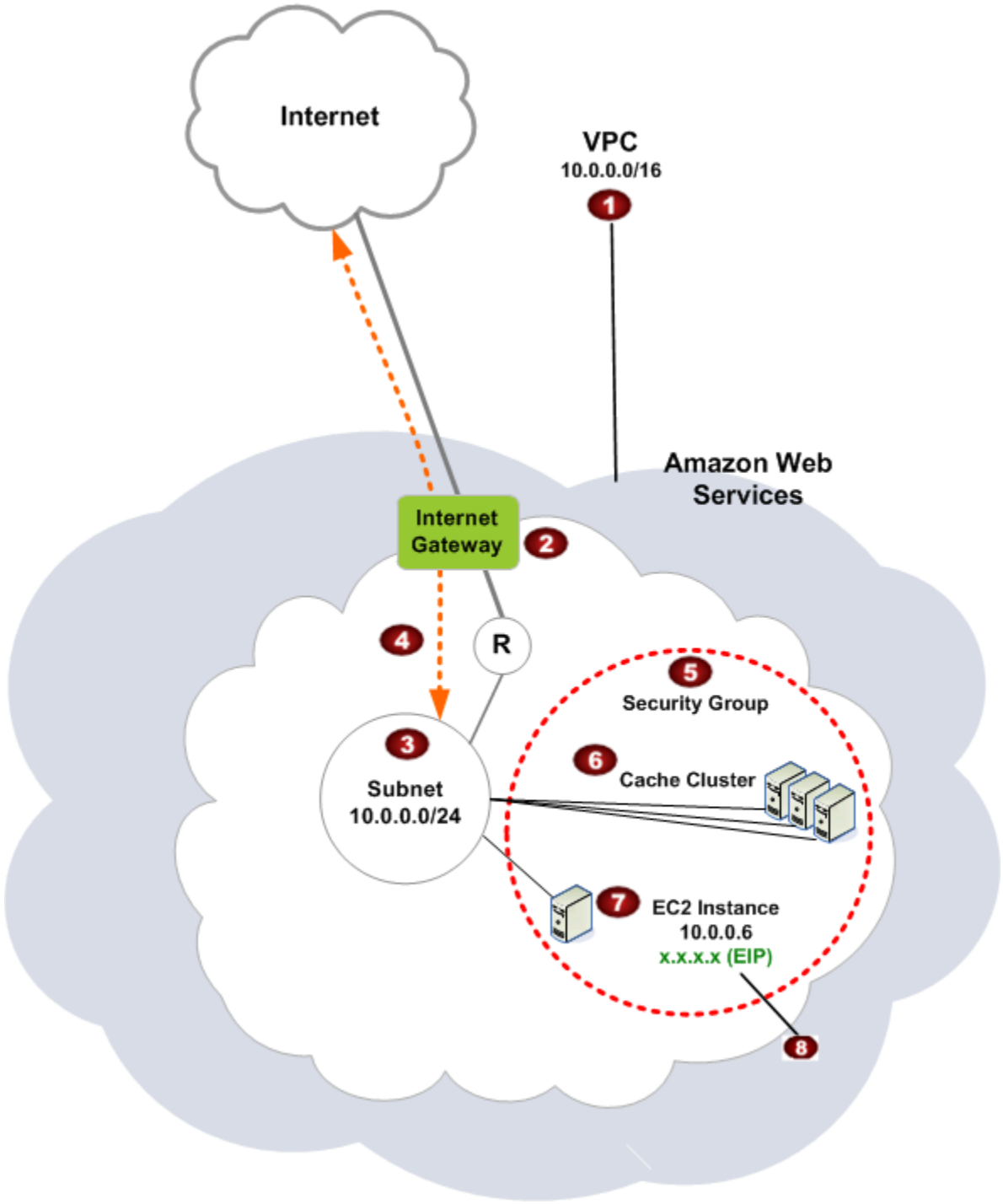
Amazon Virtual Private Cloud を使用することによって、従来のデータセンターに非常によく似た仮想ネットワークを AWS クラウド内に作成できます。お客様の Amazon VPC はお客様が設定できます。たとえば、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能です。

ElastiCache の基本機能は仮想プライベートクラウドの場合と同じです。ElastiCache は、クラスターが Amazon VPC の内部と外部のどちらにデプロイされるかに関係なく、ソフトウェアのアップグレード、パッチ適用、障害検出、および復旧を管理します。

Amazon VPC の外部にデプロイされた ElastiCache キャッシュノードには、エンドポイント/DNS 名の解決先となる IP アドレスが割り当てられます。これは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスからの接続を提供します。Amazon VPC プライベートサブネットで ElastiCache クラスターを起動した場合、すべてのキャッシュノードにはそのサブネット内のプライベート IP アドレスが割り当てられます。

Amazon VPC での ElastiCache の概要

次の図と表では、Amazon VPC 環境と、Amazon VPC で起動される ElastiCache クラスターと Amazon EC2 インスタンスについて説明します。



1	Amazon VPC は、独自の IP アドレスのブロックが割り当てられた AWS クラウドの独立した部分です。
2	

	インターネットゲートウェイは Amazon VPC を直接インターネットに接続し、他の AWS リソースへのアクセスを提供します。それには、Amazon VPC の外部で実行されている Amazon Simple Storage Service (Amazon S3) などのリソースが含まれます。
3	Amazon VPC サブネットは、セキュリティおよび運用上のニーズに合わせて AWS リソースを分離できる Amazon VPC の IP アドレス範囲のセグメントです。
4	Amazon VPC のルーティングテーブルは、サブネットとインターネットとの間でネットワークトラフィックを送信します。Amazon VPC には暗黙のルーターがあり、この図では円と R で表されています。
5	Amazon VPC セキュリティグループは、ElastiCache クラスターと Amazon EC2 インスタンスのインバウンドとアウトバウンドのトラフィックを制御します。
6	サブネットで ElastiCache クラスターを起動できます。キャッシュノードは、サブネットのアドレス範囲のプライベート IP アドレスを持ちます。
7	サブネットで Amazon EC2 インスタンスを起動することもできます。各 Amazon EC2 インスタンスはサブネットのアドレス範囲内のプライベート IP アドレスを持ちます。Amazon EC2 インスタンスは、同じサブネット内のすべてのキャッシュノードに接続できます。
8	インターネットからアクセス可能な Amazon VPC 内の Amazon EC2 インスタンスの場合は、インスタンスに Elastic IP アドレスと呼ばれる静的なパブリックアドレスを割り当てる必要があります。

前提条件

Amazon VPC 内に ElastiCache クラスターを作成するには、Amazon VPC が以下の要件を満たしている必要があります。

- Amazon VPC では、専用ではない Amazon EC2 インスタンスを許可する必要があります。ハードウェア専有インスタンスのテナンシー用に設定された Amazon VPC では ElastiCache を使用できません。

- Amazon VPC 用にキャッシュサブネットグループを定義する必要があります。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内で VPC エンドポイントまたはキャッシュノードに関連付けるサブネットおよび IP アドレスを選択します。
- 各サブネットの CIDR ブロックは、メンテナンス作業で使用する予備の IP アドレスを ElastiCache に提供するのに十分な大きさが必要です。

ルーティングとセキュリティ

Amazon VPC でルーティングを設定して、トラフィックの送信先 (インターネットゲートウェイ、仮想プライベートゲートウェイなど) を制御できます。インターネットゲートウェイの場合、Amazon VPC は、同じ Amazon VPC で実行されているのではない他の AWS リソースに直接アクセスできます。お客様の組織のローカルネットワークに接続された仮想プライベートゲートウェイのみを選択した場合、VPN 経由でインターネット宛てのトラフィックをルーティングし、ローカルセキュリティポリシーとファイアウォールを使用して送信を制御できます。この場合、インターネット経由で AWS リソースにアクセスする際に、帯域の追加料金が発生します。

Amazon VPC セキュリティグループを使用して、Amazon VPC 内の ElastiCache クラスターと Amazon EC2 インスタンスをセキュリティで保護することができます。セキュリティグループは、サブネットレベルでなくインスタンスレベルでファイアウォールのように動作します。

> **Note**
>
> 基礎となる IP アドレスは変わる可能性があるため、キャッシュノードに接続するには DNS 名を使用することを強くお勧めします。

「Amazon VPCドキュメント」

Amazon VPC に関するドキュメントには、Amazon VPC の作成および使用方法について説明する独自のドキュメントがあります。Amazon VPC ガイドへのリンクについて以下の表にまとめます。

説明	ドキュメント
Amazon VPC の使用を開始する方法	Amazon VPC の開始方法
AWS Management Console を通じて Amazon VPC を使用する方法	Amazon VPC User Guide

説明	ドキュメント
すべての Amazon VPC コマンドの詳細説明	Amazon EC2 コマンドラインリファレンス (Amazon VPC コマンドは、Amazon EC2 リファレンスに記載されています)
Amazon VPC API オペレーション、データタイプ、およびエラーの詳細説明	Amazon EC2 API リファレンス (Amazon VPC API オペレーションは、Amazon EC2 リファレンスに記載されています)
オプションとして IPsec VPN 接続のゲートウェイを設定する必要のあるネットワーク管理者向け情報	AWS Site-to-Site VPN とは

Amazon Virtual Private Cloud の詳細については、「Amazon Virtual Private Cloud」を参照してください。

Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン

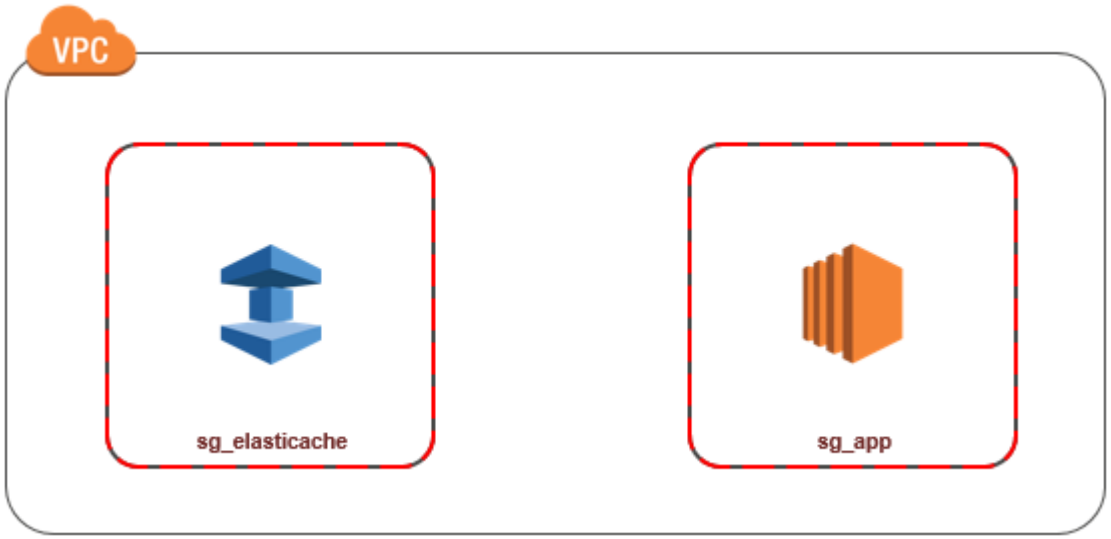
Amazon は、Amazon VPC 内のキャッシュにアクセスするための以下のシナリオ ElastiCache をサポートしています。

目次

ElastiCache キャッシュと Amazon EC2 インスタンスが同じ Amazon VPC にある場合のキャッシュへのアクセス

最も一般的ユースケースは、EC2 インスタンスにデプロイされたアプリケーションが同じ VPC のキャッシュに接続する必要がある場合です。

このシナリオを以下に図表で示します。



同じ VPC 内の EC2 インスタンスとキャッシュ間のアクセスを管理する方法として最も簡単なのは、次の方法です。

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

   Memcached キャッシュのデフォルトのポートは 11211 です。
2. EC2 インスタンス (ウェブサーバーとアプリケーションサーバー) 用の VPC セキュリティグループを作成します。このセキュリティグループは、必要に応じて VPC のルーティングテーブルを介してインターネットから EC2 インスタンスへのアクセスを許可できます。例えば、ポート 22 経由で EC2 インスタンスへの TCP アクセスを許可するルールをこのセキュリティグループに設定できます。
3. EC2 インスタンス用に作成したセキュリティグループからの接続を許可するキャッシュのセキュリティグループで、カスタムルールを作成します。これは、セキュリティグループのメンバーにキャッシュへのアクセスを許可します。

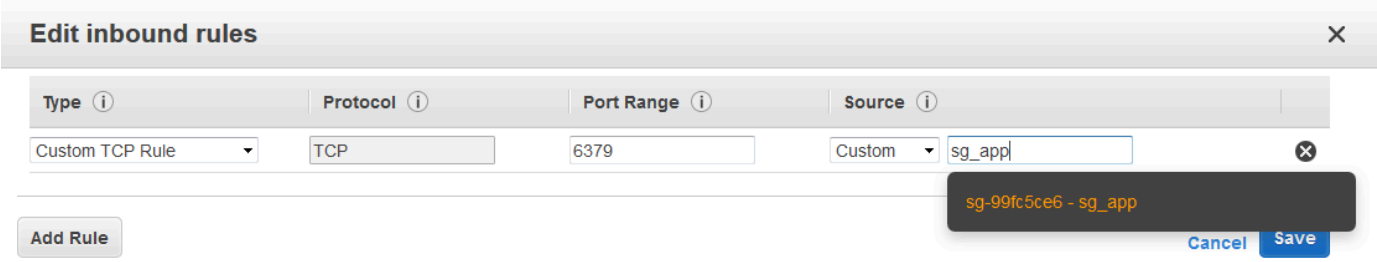
> **Note**
>
> [ローカルゾーン] の使用を予定している場合、有効になっていることを確認します。そのローカルゾーンにサブネットグループを作成すると、VPC はそのローカルゾーンに拡張さ

れ、VPC はそのサブネットを他のアベイラビリティーゾーンのサブネットとして扱います。関連するすべてのゲートウェイとルートテーブルが自動的に調整されます。

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

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

      Memcached キャッシュのデフォルトのポートは 11211 です。
   b. ソース ボックスに、セキュリティグループの ID の入力を開始します。リストから、Amazon EC2 インスタンスに使用するセキュリティグループを選択します。
5. 終了したら、保存 を選択します。



ElastiCache キャッシュと Amazon EC2 インスタンスが異なる Amazon VPCs にある場合のキャッシュへのアクセス

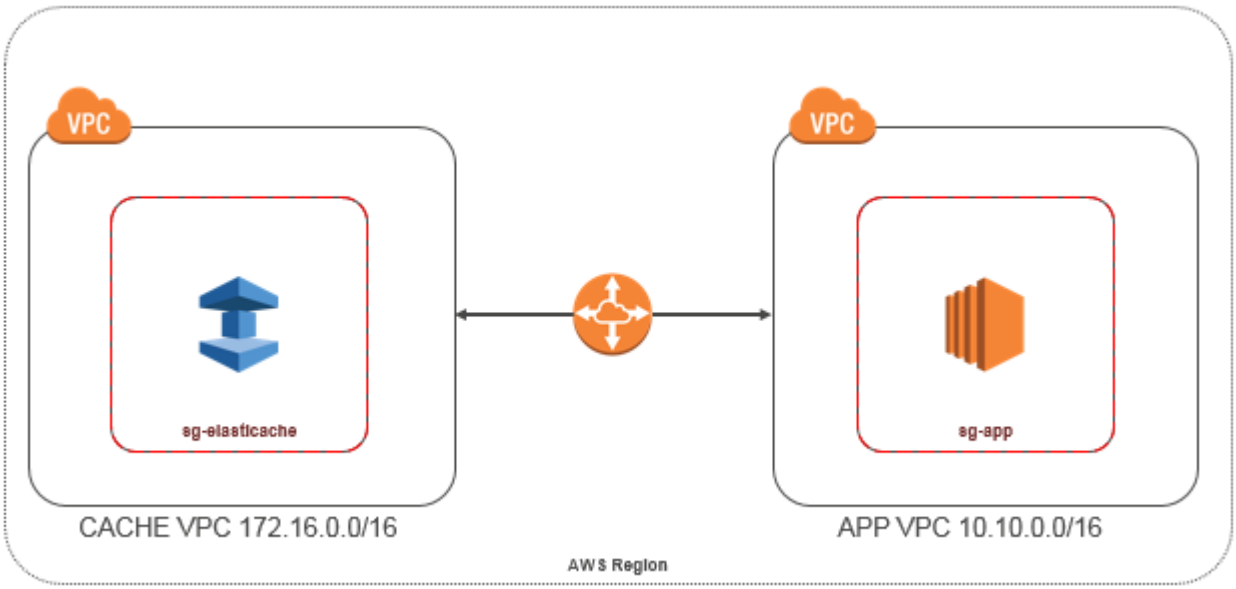
キャッシュがアクセスに使用している EC2 インスタンスとは異なる VPC にある場合、そのキャッシュにアクセスする方法はいくつかあります。キャッシュと EC2 インスタンスが異なる VPC にあるが、同じリージョンにある場合は、VPC ピアリングを使用できる。キャッシュと EC2 インスタンスが異なるリージョンにある場合、リージョン間で VPN 接続を作成できる。

トピック

- ElastiCache キャッシュと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPCs にある場合のキャッシュへのアクセス
- ElastiCache キャッシュと Amazon EC2 インスタンスが異なるリージョンの異なる Amazon VPCs にある場合のキャッシュへのアクセス

ElastiCache キャッシュと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPCs にある場合のキャッシュへのアクセス

次の図は、同じリージョンの異なる Amazon VPC での、Amazon VPC ピアリング接続を使用した、Amazon EC2 インスタンスによるキャッシュへのアクセスを示しています。



同じリージョンの異なる Amazon VPC で Amazon EC2 インスタンスによってアクセスされるキャッシュ - VPC ピアリング接続

VPC ピア接続は、プライベート IP アドレスを使用して 2 つの VPC 間でトラフィックをルーティングすることを可能にするネットワーク接続です。どちらの VPC のインスタンスも、同じネットワーク内に存在しているかのように、相互に通信できます。独自の Amazon VPC 間、または単一のリージョン内の別の AWS アカウントの Amazon VPC との間で VPCsピアリング接続を作成できます。Amazon VPC ピア接続の詳細については、「VPC ドキュメント」を参照してください。

> **Note**
>
> ElastiCache VPCsの DNS 名の解決が失敗することがあります。これを解決するには、両方の VPC を、DNS ホスト名および DNS 解決に対して有効にする必要があります。詳細については、「VPC ピアリング接続の DNS 解決を有効にする」を参照してください。

ピアリング接続経由で別の Amazon VPC のキャッシュにアクセスするには

1. 2 つの VPC に、重複する IP 範囲がないことを確認します。重複する IP 範囲がある場合、それらをピア接続することができません。
2. 2 つの VPC をピア接続します。詳細については、「VPC ピア接続の作成と使用」を参照してください。
3. ルーティングテーブルを更新します。詳細については、「VPC ピア接続のルートテーブルを更新する」を参照してください

   前述の図に示した例のルートテーブルは次のようになります。pcx-a894f1c1 はピア接続であることに注意してください。

Destination	Target
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. ElastiCache キャッシュのセキュリティグループを変更して、ピア接続された VPC のアプリケーションセキュリティグループからのインバウンド接続を許可します。詳細については、「ピア VPC セキュリティグループの参照」を参照してください。

ピアリング接続でキャッシュにアクセスすると、追加のデータ転送コストが発生します。

トランジット・ゲートウェイ の使用

トランジットゲートウェイを使用すると、同じ AWS リージョンに VPCsと VPN 接続をアタッチし、それらの間でトラフィックをルーティングできます。トランジットゲートウェイは AWS アカウント間で機能し、 AWS Resource Access Manager を使用してトランジットゲートウェイを他のア

カウントと共有できます。トランジットゲートウェイを別の AWS アカウントと共有すると、アカウント所有者は自分の VPCsをトランジットゲートウェイにアタッチできます。どちらのアカウントのユーザーも、アタッチメントをいつでも削除できます。

トランジット・ゲートウェイ でマルチキャストを有効にしてから、ドメインに関連付ける VPC アタッチメントを介してマルチキャストソースからマルチキャストグループメンバーにマルチキャストトラフィックを送信できるようにする トランジット・ゲートウェイ マルチキャストドメインを作成できます。

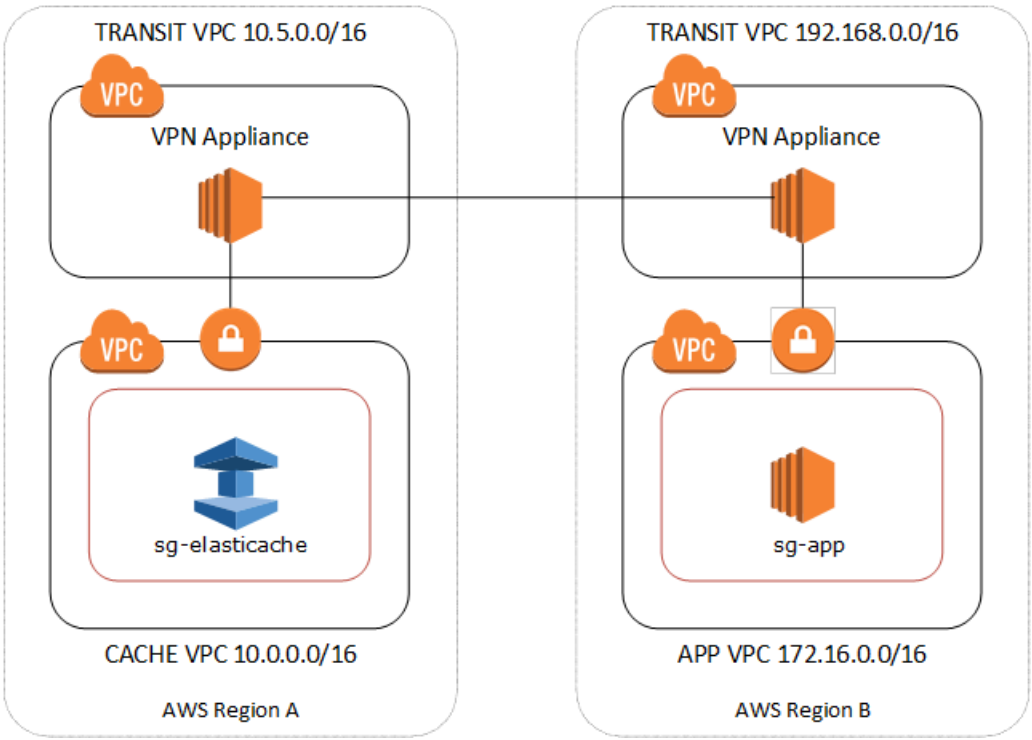
異なる AWS リージョンのトランジットゲートウェイ間にピアリング接続アタッチメントを作成することもできます。これにより、異なるリージョン間でトランジット・ゲートウェイのアタッチメント間でトラフィックをルーティングできます。

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の異なる Amazon VPCs にある場合のキャッシュへのアクセス

トランジット VPC の使用

VPC ピアリングの代わりに使用する、複数の、地理的に離れた VPC とリモートネットワークを接続する別の一般的な方法は、グローバルなネットワーク中継センターとして機能する中継 VPC の作成です。中継 VPC はネットワーク管理を単純化して、複数の VPC とリモートのネットワークを接続するために必要な接続数を最小限に抑えます。この設計は、コロケーション中継ハブを物理的に設立したり、物理的なネットワーク設備をデプロイしたりするための従来の費用をほとんどかけずに実装できるため、時間と労力を節約し、コストも削減できます。



異なるリージョンの異なる VPC 間での接続

Transit Amazon VPC が確立されると、あるリージョンの「スポーク」VPC にデプロイされたアプリケーションは、別のリージョン内の「スポーク」VPC の ElastiCache キャッシュに接続できます。

別の AWS リージョン内の別の VPC のキャッシュにアクセスするには

1. Transit VPC ソリューションをデプロイします。詳細については、「AWS Transit Gateway」を参照してください。
2. アプリおよびキャッシュ VPC の VPC ルーティングテーブルを更新し、VGW (仮想プライベートゲートウェイ) および VPN アプライアンスを通じてトラフィックをルーティングします。ボーダーゲートウェイプロトコル (BGP) を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. ElastiCache キャッシュのセキュリティグループを変更して、アプリケーションインスタンスの IP 範囲からのインバウンド接続を許可します。このシナリオでは、アプリケーションサーバーセキュリティグループを参照することはできません。

リージョン間でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、リージョン間のデータ転送コストが追加で発生します。

お客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

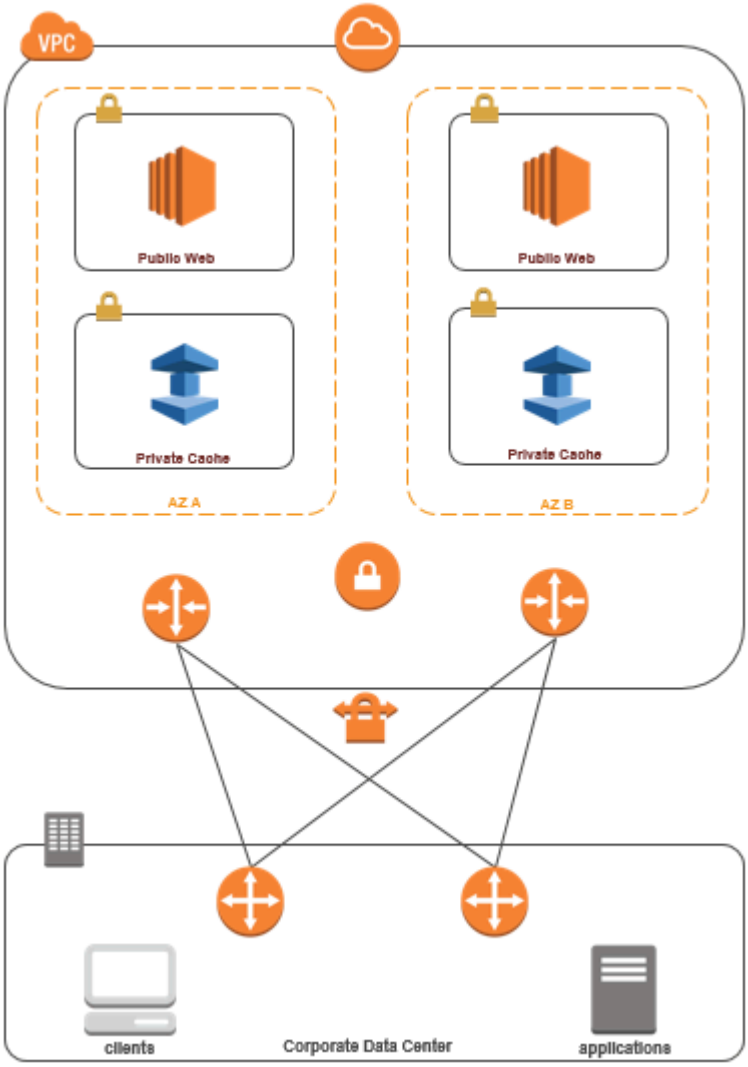
もう 1 つのシナリオは、顧客のデータセンター内のクライアントまたはアプリケーションが VPC 内の ElastiCache キャッシュにアクセスする必要があるハイブリッドアーキテクチャです。このシナリオは、顧客の VPC とデータセンター間で VPN または Direct Connect による接続がある場合にサポートされます。

トピック

- VPN 接続を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする
- Direct Connect を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

VPN 接続を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

次の図は、VPN 接続を使用して企業ネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスする方法を示しています。



VPN 経由でデータセンター ElastiCache から に接続する

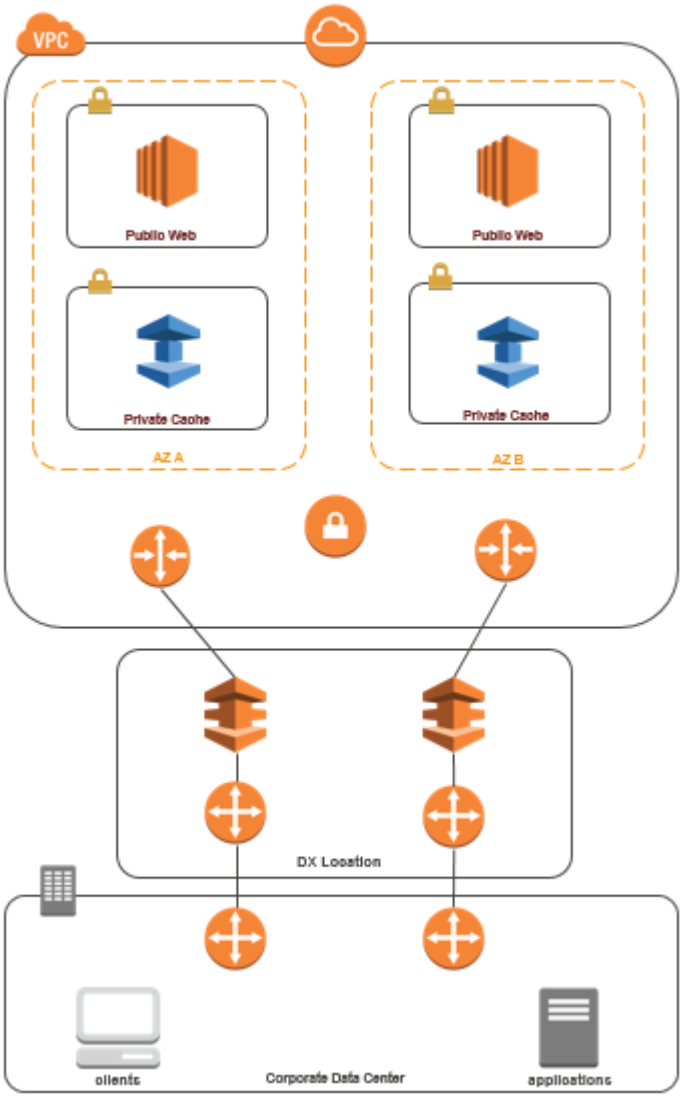
VPN 接続経由でオンプレミスアプリケーションから VPC のキャッシュにアクセスするには

1. VPC にハードウェア仮想プライベートゲートウェイを追加して、VPN 接続を確立します。詳細については、「VPC へのハードウェア仮想プライベートゲートウェイの追加」を参照してください。
2. ElastiCache キャッシュがデプロイされているサブネットの VPC ルーティングテーブルを更新して、オンプレミスアプリケーションサーバーからのトラフィックを許可します。BGP を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. ElastiCache キャッシュのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

VPN 接続経由でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送コストが発生します。

Direct Connect を使用してお客様のデータセンターで実行されているアプリケーションから ElastiCache キャッシュにアクセスする

次の図は、Direct Connect を使用して企業ネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスする方法を示しています。



Direct Connect 経由でデータセンター ElastiCache から に接続する

Direct Connect を使用してネットワークで実行されているアプリケーションから ElastiCache キャッシュにアクセスするには

1. Direct Connect 接続を確立します。詳細については、AWS 「Direct Connect の開始方法」を参照してください。
2. ElastiCache キャッシュのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

DX 接続経由でキャッシュにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送料金が発生する場合があります。

Virtual Private Cloud (VPC) の作成

この例では、各アベイラビリティーゾーンのプライベートサブネットを持つ Amazon VPC を作成します。

Amazon VPC の作成 (コンソール)

1. AWS マネジメントコンソールにサインインして Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. VPC ダッシュボードで、Create VPC (VPC の作成) を選択します。
3. Resources to create (作成するリソース) で、VPC only (VPC など) を選択します。
4. Number of Availability Zones (AZs) (アベイラビリティゾーンの数 (AZ)) で、サブネットを起動するアベイラビリティーゾーンの数を選択します。
5. Number of public subnets (パブリックサブネットの数) で、VPC に追加するパブリックサブネットの数を選択します。
6. Number of private subnets (プライベートサブネットの数) で、VPC に追加するプライベートサブネットの数を選択します。

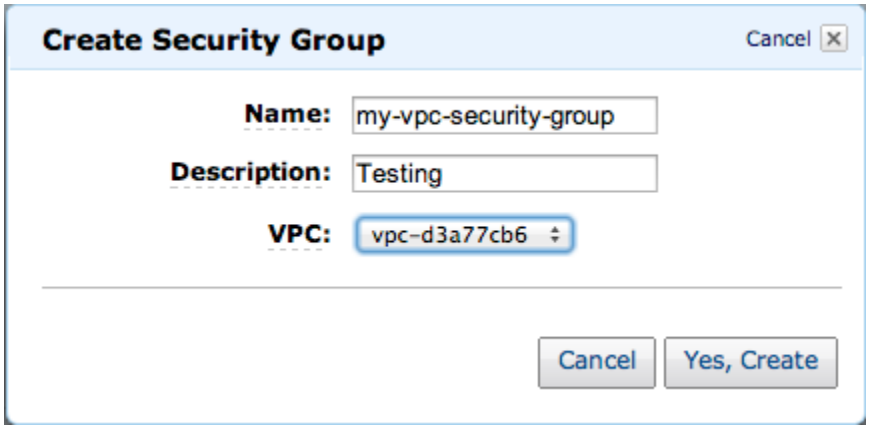
   > **Tip**
   >
   > サブネットの識別子と、どちらがパブリックで、どちらがプライベートであるかを書き留めておきます。この情報は、後でクラスターを起動し、Amazon VPC に Amazon EC2 インスタンスを追加するときに必要になります。

7. Amazon VPC セキュリティグループを作成します。キャッシュクラスターと Amazon EC2 インスタンスでは、このグループを使用します。

   a. Amazon VPC Management Console のナビゲーションペインで、[Security Group (セキュリティグループ)] を選択します。

   b. [Create Security Group (セキュリティグループの作成)] を選択します。

   c. 対応するボックスにセキュリティグループの名前と説明を入力します。[VPC] ボックスで Amazon VPC の ID を選択します。



d. すべての設定が正しいことを確認したら、Yes, Create を選択します。

8. セキュリティグループのネットワーク Ingress ルールを定義します。このルールは、Secure Shell (SSH) を使用して Amazon EC2 インスタンスに接続することを許可します。

   a. ナビゲーションリストで [Security Groups] を選択します。
   b. リストで対象となるセキュリティグループを探して選択します。
   c. Security Group の下で、Inbound タブを選択します。Create a new rule ボックスで、SSH を選択し、Add Rule を選択します。
   d. 新しいインバウンドルールに次の値を設定して、HTTP へのアクセスを許可します。

      - Type: HTTP
      - ソース: 0.0.0.0/0

      Apply Rule Changes を選択します。

これで、キャッシュサブネットグループを作成して Amazon VPC でキャッシュクラスターを起動する準備が整いました。

- サブネットグループの作成
- Memcached クラスター (CLI) の作成 (コンソール).

Amazon VPC で実行されるキャッシュへの接続

この例では、Amazon VPC で Amazon EC2 インスタンスを起動する方法を示します。その後、このインスタンスにログインし、Amazon VPC で実行されている ElastiCache キャッシュにアクセスできます。

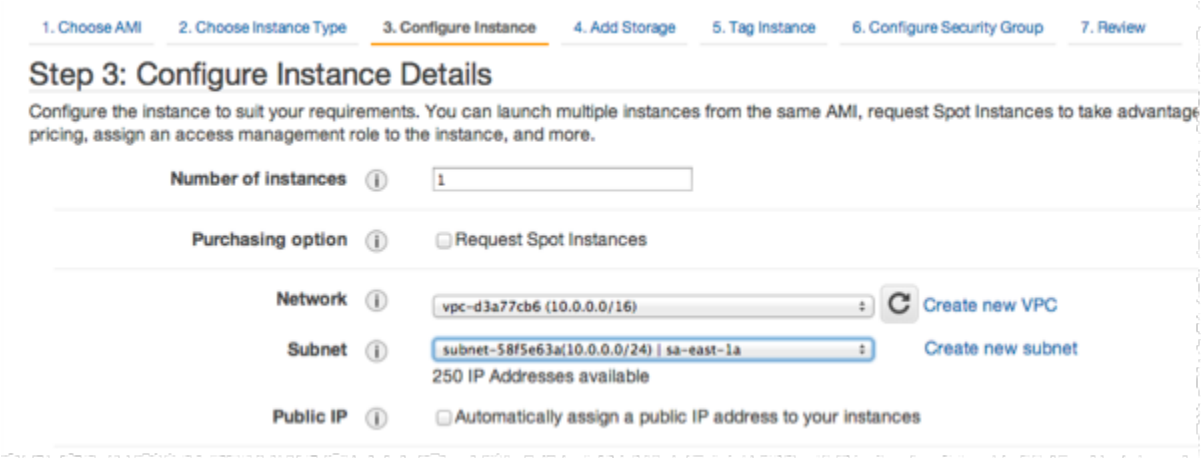
Amazon VPC で実行されるキャッシュへの接続 (コンソール)

この例では、Amazon VPC で Amazon EC2 インスタンスを作成します。この Amazon EC2 インスタンスを使用して、Amazon VPC で実行中のキャッシュノードに接続できます。

> **Note**
>
> Amazon EC2 の使用に関する詳細は、Amazon EC2 ドキュメントの「Amazon EC2 入門ガイド」を参照してください。

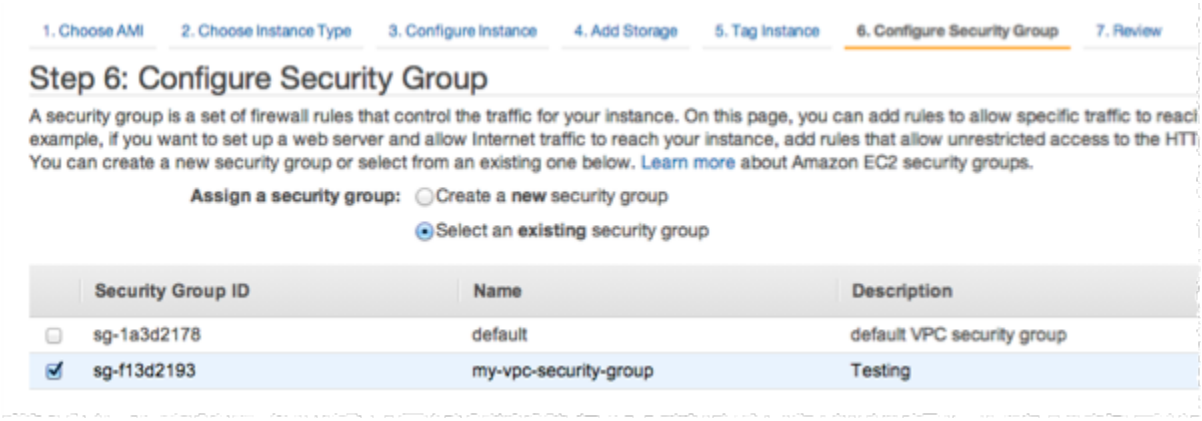
Amazon EC2 コンソールを使用して Amazon VPC で Amazon EC2 インスタンスを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. 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 (AMI)] ページで、64 ビット Amazon Linux AMI を選択し、[選択] をクリックします。
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 VPC を選択します。
   b. [サブネット] リストで、パブリックサブネットを選択します。



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. [インスタンスのタグ付け] ページで、Amazon EC2 インスタンスの名前を入力し、[6. 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 インスタンス用の 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、[Review and Launch (確認および起動)] 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**
>
> キーペアの管理の詳細については、「Amazon EC2 入門ガイド」を参照してください。

11. Amazon EC2 インスタンスを起動する準備ができたら、[起動] を選択します。

これで、先ほど作成した Amazon EC2 インスタンスにElastic IP アドレスを割り当てることができます。この IP アドレスは、Amazon EC2 インスタンスに接続するときに使用する必要があります。

Elastic IP アドレスを割り当てるには (コンソール)

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションリストで [Elastic IP] を選択します。
3. [Elastic IP アドレスを割り当てる] 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。

   これで、SSH を使用して、作成した Elastic IP アドレスを使用して Amazon EC2 インスタンスに接続できるようになりました。

Amazon EC2 インスタンスに接続するには

- コマンドウィンドウを開きます。コマンドプロンプトで、次のコマンドを実行します。mykeypair.pem をご使用のキーペアファイルの名前に、54.207.55.251 をご使用の Elastic IP アドレスに置き換えます。

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

> **Important**
>
> まだ Amazon EC2 インスタンスからログアウトしないでください。

これで、 ElastiCache クラスターを操作する準備ができました。まだ telnet ユーティリティをインストールしていない場合、これを行うには、このユーティリティをインストールする必要があります。

telnet をインストールし、キャッシュクラスター (AWS CLI) を操作するには

1. コマンドウィンドウを開きます。コマンドプロンプトで、次のコマンドを発行します。確認のプロンプトが表示されたら、「y」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!
```

2. https://console.aws.amazon.com/elasticache/ の ElastiCache コンソールに移動し、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、Memcached の「接続エンドポイントの検索」を参照してください。
3. 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
```

> **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は、不要になったリソースをすべて削除 AWS してください。

Amazon ElastiCache API とインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、VPC と Amazon ElastiCache API エンドポイント間にプライベート接続を確立できます。インターフェイスエンドポイントはAWS PrivateLinkを使用します。AWS PrivateLinkを使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで、Amazon ElastiCache API オペレーションにプライベートにアクセスできます。

VPC のインスタンスは、パブリック IP アドレスがなくても Amazon ElastiCache API エンドポイントと通信できます。また、ElastiCache API オペレーションの使用にも、パブリック IP アドレスを必要としません。VPC と Amazon ElastiCache 間のトラフィックは、Amazon ネットワークを離れません。各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。Elastic Network Interface の詳細については、Amazon EC2 ユーザーガイドの「Elastic Network Interface」を参照してください。

- VPC エンドポイントの詳細については、Amazon VPC ユーザーガイドの「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。
- ElastiCache API オペレーションの詳細については、「ElastiCache API オペレーション」を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にすると、デフォルトの ElastiCache エンドポイント (https://elasticache.*Region*.amazonaws.com) はお客様の VPC エンドポイントに解決されます。プライベート DNS ホスト名を有効にしない場合は、Amazon VPC が以下の形式で使用できる 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 VPC ユーザーガイドの「AWSインターフェイス VPC エンドポイント (PrivateLink)」をご参照ください。ElastiCache は、VPC 内のすべての API アクションへの呼び出しをサポートしています。

> **Note**
>
> プライベート DNS ホスト名は、VPC 内の 1 つの VPC エンドポイントに対してのみ有効にできます。追加の VPC エンドポイントを作成する場合は、プライベート DNS ホスト名を無効にする必要があります。

VPC エンドポイントに関する考慮事項

Amazon ElastiCache API エンドポイントのインターフェイス VPC エンドポイントを設定する前に、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントのプロパティと制限」を確認してください。Amazon ElastiCache リソースの管理に関連するすべての ElastiCache API オペレーションは、AWS PrivateLink を使用して VPC から利用することができます。

VPC エンドポイントポリシーは ElastiCache API エンドポイントでサポートされます。デフォルトでは、エンドポイント経由で ElastiCache API オペレーションへのフルアクセスが許可されます。詳細については、Amazon VPC ユーザーガイドのVPC エンドポイントによるサービスのアクセスコントロールを参照してください。

ElastiCache API 用のインターフェイス VPC エンドポイントの作成

Amazon ElastiCache API 用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS CLI で作成できます。詳細については、Amazon VPC ユーザーガイドのインターフェイスエンドポイントの作成を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にできます。その場合、デフォルトの Amazon ElastiCache エンドポイント (https://elasticache.*Region*.amazonaws.com) は、お客様の VPC エンドポイントに解決されます。中国 (北京) および 中国 (寧夏) AWS リージョンの場合、北京に `elasticache.cn-north-1.amazonaws.com.cn`、寧夏に `elasticache.cn-northwest-1.amazonaws.com.cn` を使用して VPC エンドポイントで API リクエストを行うことができます。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

Amazon ElastiCache API 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに ElastiCache API へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。本ポリシーでは、以下を規定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイドの「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

Example ElastiCache API アクションの VPC エンドポイントポリシー

ElastiCache API のエンドポイントポリシーの例を次に示します。このポリシーは、エンドポイントにアタッチされると、すべてのリソースのすべてのプリンシパルに対して、登録されている ElastiCache API アクションへのアクセスを許可します。

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster"
  ],
  "Resource": "*"
 }]
}
```

Example 指定した AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、AWS アカウント *123456789012* がエンドポイントを使用するリソースへのすべてのアクセスを拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
```

```
    "Action": "*",
    "Effect": "Deny",
    "Resource": "*",
    "Principal": {
     "AWS": [
      "123456789012"
     ]
    }
   }
  ]
}
```

サブネットおよびサブネットグループ

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

Amazon VPC で独自設計型クラスターを作成する場合、サブネットグループを使用する必要があります。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。

ElastiCache はデフォルトの IPv4 サブネットグループを提供しています。または、新しいサブネットグループを作成することもできます。IPv6 の場合は、IPv6 CIDR ブロックを使用するサブネットグループを作成する必要があります。デュアルスタックを選択した場合は、[Discovery IP type] (検出 IP タイプ) (IPv6 または IPv4) を選択する必要があります。

ElastiCache サーバーレスはサブネットグループリソースを使用せず、代わりに作成時にサブネットのリストを直接取得します。

このセクションでは、サブネットおよびサブネットグループを作成し活用して、ElastiCache リソースへのアクセスを管理する方法を扱います。

Amazon VPC 環境でのサブネットグループの使用方法の詳細については、「クラスターへのアクセス」を参照してください。

トピック

- サブネットグループの作成
- キャッシュにサブネットグループを割り当てる
- サブネットグループの変更
- サブネットグループの削除

サブネットグループの作成

キャッシュサブネットグループは、VPC 内でとして指定できるサブネットの集合です。VPC でキャッシュを起動する場合は、キャッシュサブネットグループを選択する必要があります。次に、ElastiCache ではそのキャッシュサブネットグループを使用して、サブネット内の IP アドレスをキャッシュ内の各キャッシュノードに割り当てます。

新しいサブネットグループを作成する場合は、使用可能な IP アドレス数に注意してください。サブネットの空き IP アドレス数が非常に少ない場合は、クラスターに追加できるノード数が制約される可能性があります。この問題を解決するために、クラスターのアベイラビリティ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として IPv4 を選択した場合は、デフォルトのサブネットグループが利用でき、新しいサブネットグループを作成することもできます。ElastiCache はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。デュアルスタックまたは IPV6 を選択すると、デュアルスタックまたは IPV6 サブネットを作成するように指示されます。ネットワークタイプの詳細については、「ネットワークタイプ」を参照してください。詳細については、「VPC にサブネットを作成する」を参照してください。

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

サブネットグループの作成 (コンソール)

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

サブネットグループ (コンソール) を作成するには

1. AWS マネジメントコンソールにサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. ナビゲーションリストで [サブネットグループ] を選択します。
3. [サブネットグループの作成] を選択します。
4. [サブネットグループを作成] ウィザードで、次の操作を行います。すべての設定が正しいことを確認したら、作成 (作成) を選択します。

    a. Name ボックスにサブネットグループの名前を入力します。
    b. Description ボックスにサブネットグループの説明を入力します。
    c. [VPC ID] ボックスで Amazon VPC を選択します。

d. デフォルトでは、すべてのサブネットが選択されています。[選択済みのサブネット] パネルで [管理] をクリックし、プライベートサブネットのアベイラビリティーゾーンまたはローカルゾーンと ID を選択し、[選択] をクリクします。

5. 表示された確認メッセージで、Close を選択します。

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

サブネットグループの作成 (AWS CLI)

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

Linux、macOS、Unix の場合:

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

Windows の場合:

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

このコマンドでは、次のような出力が生成されます。

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

```
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

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

キャッシュにサブネットグループを割り当てる

サブネットグループを作成したら、Amazon VPC でキャッシュを起動できます。詳細については、以下を参照してください。

- [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 ([Memcached の詳細設定]) 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ことや、サブネットグループに関連付けられたサブネット ID のリストを変更することができます。キャッシュが現在サブネットを使用している場合、サブネットグループからそのサブネット ID を削除することはできません。

次の手順では、サブネットグループを変更する方法を示します。

サブネットグループの変更 (コンソール)

サブネットグループを変更するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. [ナビゲーション] ペインで、[サブネットグループ] を選択します。
3. サブネットグループのリストで、変更する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. [選択済みのサブネット]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 のトピック「modify-cache-subnet-group」を参照してください。

サブネットグループの削除

サブネットグループが必要ではなくなったと判断した場合、サブネットグループを削除できます。サブネットグループがキャッシュで現在使用されている場合、サブネットグループを削除できません。

次の手順では、サブネットグループを削除する方法を示します。

サブネットグループの削除 (コンソール)

サブネットグループを削除するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. [ナビゲーション] ペインで、[サブネットグループ] を選択します。
3. サブネットグループのリストで、削除する項目を選択して [Delete] を選択します。
4. この操作を確定するように求められたら、テキスト入力フィールドにサブネットグループの名前を入力し、[削除] を選択します。

サブネットグループの削除 (AWS CLI)

AWS CLI を使用して、以下のパラメーターでコマンド delete-cache-subnet-group を呼び出します。

- `--cache-subnet-group-name` *`mysubnetgroup`*

Linux、macOS、Unix の場合:

```
aws elasticache delete-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup
```

Windows の場合:

```
aws elasticache delete-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup
```

このコマンドでは何も出力されません。

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

Amazon の Identity and Access Management ElastiCache

AWS Identity and Access Management （ IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS サービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に ElastiCache リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は追加料金なしで AWS 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、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの ElastiCache 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、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、通常、 へのフルアクセスがあります ElastiCache。サービスユーザーがどの ElastiCache 機能やリソースにアクセスするかを決めるのは管理者の仕事です。次に、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てくださいIAM。会社で を と使用する方法の詳細については、IAM 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ポリシーの作成方法の詳細について確認する場合があります ElastiCache。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 ElastiCache のアイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAMユーザーとして AWS アカウントのルートユーザー、または IAMロールを引き受けることによって認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center (IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインすると、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、「 ユーザーガイド」の「 にサインインする方法 AWS アカウントAWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は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で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、「 IAMユーザーガイド」のAWS API「リクエストの署名」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用することをお勧めします。詳細については、「 AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「 ユーザーガイド」の「 での多要素認証 (MFA) AWS IAM の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS サービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実

行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「 IAMユーザーガイド」の「ルートユーザー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、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするために ID プロバイダーとのフェデレーションを使用することを要求 AWS サービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS サービス を使用して にアクセスするユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「 ユーザーガイド」のIAM「Identity Center とはAWS IAM Identity Center 」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を持つIAMユーザーを作成するのではなく、一時的な認証情報を使用することをお勧めします。ただし、IAMユーザー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、「 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でアクセスキーを定期的にローテーションするIAM」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アイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、 という名前のグループを作成し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、「 ユーザーガイド」のIAM「 (ロールの代わりに) ユーザーを作成する場合IAM」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。ユーザーと似ていますがIAM、特定のユーザーに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。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム を使用しますURL。ロールの使用方法の詳細については、「 ユーザーガイド」のIAM「ロールの使用IAM」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、以下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、「 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IAM」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、 IAMロールを引き受けて、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、別のアカウントのユーザー (信頼されたプリンシパル) 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 サービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「 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IAMIAM」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS サービス を使用します AWS 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を AWS サービス、ダウンストリームサービス AWS サービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS 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、「転送アクセスセッション」を参照してください。
- サービスロール – サービスロールは、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、「 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 サービスIAM」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 サービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

- 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、「 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IAM」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、「 ユーザーガイド」の「 (ユーザーではなく) IAMロールを作成する場合IAM」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション)

AWS がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーはJSONドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「 ユーザーガイド」のJSON「ポリシーの概要IAM」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ために、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メソッドに関係なく、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。アイデンティティベースのポリシーを作成する方法については、「 ユーザーガイド」のIAM「ポリシーの作成IAM」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。管理ポリシーとインラインポリシーのどちらかを選択する方法については、「 IAM ユーザーガイド」の「管理ポリシーとインラインポリシーの選択」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ロールの信頼ポリシー や Amazon S3 バケットポリシー などがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを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 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、 の AWS 管理ポリシーを使用できません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) 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を使用しません。

Amazon S3、 AWS WAF、および Amazon VPCは、 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) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 – アクセス許可の境界は、アイデンティティベースのポリシーがIAMエンティティ (IAMユーザーまたはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「 IAMユーザーガイド」の「 IAMエンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) に対する最大アクセス許可を指定するJSONポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCPs) をアカウントの一部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IAM」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、 ユーザーガイドの「ポリシー評価ロジックIAM」を参照してください。

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機能を確認してください ElastiCache。

IAM Amazon で使用できる の機能 ElastiCache

IAM 機能	ElastiCache サポート
アイデンティティベースのポリシー	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	はい
ACLs	あり
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	あり

IAM 機能	ElastiCache サポート
サービスリンクロール	あり

ElastiCache およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、 IAM ユーザーガイドのAWS 「 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。アイデンティティベースのポリシーを作成する方法については、「 ユーザーガイド」のIAM「ポリシーの作成IAM」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否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、「 ユーザーガイド」の「 IAMJSONポリシー要素のリファレンスIAM」を参照してください。

ElastiCache のアイデンティティベースのポリシーの例

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon ElastiCache のアイデンティティベースのポリシーの例。

ElastiCache 内のリソースベースのポリシー

リソースベースのポリシーのサポート: 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ロールの信頼ポリシー や Amazon S3 バケットポリシー などがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを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 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、アカウント全体または別のアカウントのIAMエンティティを指定できます。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、「 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IAMIAM」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、ポリシーで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の名前は通常、関連する AWS APIオペレーションと同じです。一致するAPIオペレーション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 で定義されるアクション ElastiCache」を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス ElastiCache を使用します。

```
elasticache
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "elasticache:action1",
```

```
      "elasticache:action2"
         ]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "elasticache:Describe*"
```

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon ElastiCache のアイデンティティベースのポリシーの例。

のポリシーリソース ElastiCache

ポリシーリソースのサポート: 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。

Policy ResourceJSON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 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。

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon ElastiCache のアイデンティティベースのポリシーの例。

ElastiCache 向けのポリシー条件キー

サービス固有のポリシー条件キーのサポート: 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、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アクセス許可をIAMユーザーに付与できますIAM。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IAM」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「 ユーザーガイド」のAWS 「 グローバル条件コンテキストキーIAM」を参照してください。

ElastiCache 条件キーのリストを確認するには、「サービス認証リファレンス」の「Amazon の条件キー ElastiCache」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon で定義されるアクション ElastiCache」を参照してください。

ElastiCache アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon ElastiCache のアイデンティティベースのポリシーの例。

のアクセスコントロールリスト (ACLs) ElastiCache

をサポートACLs： はい

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) 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を使用しません。

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 (ポリシー内のタグ): 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可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAMエンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、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場合に、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や、ポリシー管理が煩雑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。のセットアップ手順を含むチュートリアルを表示するにはABAC、「 ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用するIAM」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

一部の は、一時的な認証情報を使用してサインインすると機能 AWS サービス しません。一時的な認証情報 AWS サービス を使用する などの詳細については、 ユーザーガイドのAWS サービス 「 と連携する IAM 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。

一時的な認証情報は、 AWS CLI または を使用して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 . AWS recommends にアクセスできます AWS。これは、長期的なア

クセスキーを使用する代わりに、一時的な認証情報を動的に生成することを推奨しています。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を AWS サービス、ダウンストリームサービス AWS サービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS 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、「転送アクセスセッション」を参照してください。

のサービスロール ElastiCache

サービスロールのサポート: 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、「 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 サービスIAM」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、 ElastiCache 機能が破損する可能性があります。が指示する場合以外 ElastiCache は、サービスロールを編集しないでください。

のサービスにリンクされたロール ElastiCache

サービスリンクロールのサポート: 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 サービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「AWS と連携する のサービス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 ElastiCache のアイデンティティベースのポリシーの例

デフォルトで、ユーザーとロールには ElastiCache リソースを作成または変更する許可がありません。また、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者がロールに IAM ポリシーを追加すると、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

ElastiCache が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認証リファレンス」の「Amazon ElastiCache のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- ElastiCache コンソールの使用
- 自分の権限の表示をユーザーに許可する

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰が ElastiCache リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、権限をさらに減らすことをお勧めし

ます。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。

- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS CloudFormation などの特定の AWS サービス を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、『IAM ユーザーガイド』の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する - AWS アカウント内の IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

ElastiCache コンソールの使用

Amazon ElastiCache コンソールにアクセスするには、アクセス許可の最小限のセットが必要です。これらのアクセス許可により、AWS アカウント の ElastiCache リソースの詳細をリストおよび表示できます。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き ElastiCache コンソールを使用できるようにするには、エンティティに ElastiCache `ConsoleAccess` または `ReadOnly` AWS 管理ポリシーもアタッチします。詳細については、『IAM ユーザーガイド』の「ユーザーへの権限の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam: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

次の情報は、 と IAM の使用時に発生する可能性がある一般的な問題の診断 ElastiCache と修正に役立ちます。

トピック

- でアクションを実行する権限がない ElastiCache
- iam を実行する権限がありません。PassRole
- 自分の AWS アカウント以外のユーザーに自分の ElastiCache リソースへのアクセスを許可したい

でアクションを実行する権限がない ElastiCache

からアクションを実行する権限がないと AWS Management Console 通知された場合は、管理者に連絡してサポートを依頼する必要があります。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

以下のエラー例は、mateojackson ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の elasticache:*GetWidget* 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticache:GetWidget on resource: my-example-widget
```

この場合、Mateo は、elasticache:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスが許可されるように、管理者にポリシーの更新を依頼します。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して にロールを渡すことができるようにする必要があります ElastiCache。

一部の AWS サービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、 という IAM ユーザーがコンソールを使用して marymajor でアクションを実行しようする場合に発生します ElastiCache。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント以外のユーザーに自分の ElastiCache リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 ElastiCache をサポートしているかどうかを確認するには、「」を参照してくださいAmazon と 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。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部認証されたユーザーへのアクセスの提供 (ID フェデレーション) を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

アクセスコントロール

リクエストを認証するために有効な認証情報を持つことができますが、アクセス許可がない限り、ElastiCache リソースを作成またはアクセスすることはできません。例えば、 ElastiCache クラスターを作成するアクセス許可が必要です。

以下のセクションでは、 のアクセス許可を管理する方法について説明します ElastiCache。最初に概要のセクションを読むことをお勧めします。

- ElastiCache リソースに対するアクセス許可の管理の概要
- Amazon ElastiCache でのアイデンティティベースのポリシー (IAM ポリシー) の使用

ElastiCache リソースに対するアクセス許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースの作成またはアクセスは、許可のポリシーによって管理されます。アカウント管理者は、IAM アイデンティティ (ユーザー、グループ、ロール) にアクセス許可ポリシーをアタッチできます。さらに、Amazon ElastiCache では、アクセス許可ポリシーをリソースにアタッチすることもできます。

> **Note**
>
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、「IAM ユーザーガイド」の「IAM のベストプラクティス」を参照してください。

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

- AWS IAM Identity Center のユーザーとグループ:

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「アクセス許可一式を作成」の手順を実行します。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

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

トピック

- Amazon ElastiCache のリソースとオペレーション
- リソース所有権について
- リソースへのアクセスの管理
- Amazon ElastiCache 用の AWS マネージドポリシー
- Amazon ElastiCache でのアイデンティティベースのポリシー (IAM ポリシー) の使用

- リソースレベルのアクセス許可
- 条件キーの使用
- Amazon ElastiCache でのサービスにリンクされたロールの使用
- ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス

Amazon ElastiCache のリソースとオペレーション

ElastiCache リソースのタイプとその ARN のリストを確認するには、「サービス認証リファレンス」の「Amazon ElastiCache で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon ElastiCache で定義されるアクションを参照してください。

リソース所有権について

リソース所有者は、リソースを作成した AWS アカウントです。つまり、リソース所有者は、そのリソースを作成するリクエストを認証するプリンシパルエンティティの AWS アカウントです。プリンシパルエンティティ はルートアカウント、IAM ユーザー、または IAM ロールです。次の例は、この仕組みを示しています。

- AWS アカウントのルートアカウント認証情報を使用してキャッシュクラスターを作成するとします。この場合、AWS アカウントはリソースの所有者です。ElastiCache では、リソースはキャッシュクラスターです。
- AWS アカウントに IAM ユーザーを作成し、キャッシュクラスターを作成するためのアクセス許可をそのユーザーに付与するとします。この場合、ユーザーはキャッシュクラスターを作成できます。ただし、キャッシュクラスターリソースを所有しているのは、このユーザーが属する AWS アカウントです。
- キャッシュクラスターを作成するためのアクセス許可のある AWS アカウントに IAM ロールを作成するとします。この場合、ロールを引き受けることができるいずれのユーザーもキャッシュクラスターを作成できます。ロールが属する AWS アカウントがキャッシュクラスターリソースを所有しています。

リソースへのアクセスの管理

アクセス権限ポリシー では、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス許可ポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、Amazon ElastiCache のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。完全な IAM ドキュメンテーションについては、「IAM ユーザーガイド」の「IAM とは」を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの AWS IAM ポリシーの参照を参照してください。

IAM アイデンティティにアタッチされているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースに添付されたポリシーは、リソースベースのポリシーと呼ばれます。

トピック

- アイデンティティベースのポリシー (IAM ポリシー)
- ポリシー要素の指定: アクション、効果、リソース、プリンシパル
- ポリシーでの条件の指定

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。例えば、次のオペレーションを実行できます。

- アカウントのユーザーまたはグループにアクセス許可ポリシーをアタッチする – アカウント管理者は、特定のユーザーに関連付けられるアクセス許可ポリシーを使用して、アクセス許可を付与できます。この場合、アクセス許可は、そのユーザーがキャッシュクラスター、パラメータグループ、セキュリティグループなどの ElastiCache リソースを作成するためのものです。
- 許可ポリシーをロールにアタッチする (クロスアカウントの許可を付与) - ID ベースのアクセス許可ポリシーを IAM ロールにアタッチして、クロスアカウントの許可を付与することができます。例えば、アカウント A の管理者は、次のように別の AWS アカウント (例えば、アカウント B) または AWS サービスにクロスアカウントアクセス許可を付与するロールを作成できます。
  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースにアクセス許可を付与するロールにアクセス許可ポリシーをアタッチします。
  2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。

3. アカウント B の管理者は、ロールを引き受けるためのアクセス許可をアカウント B のユーザーに委任できます。これにより、アカウント B のユーザーは、アカウント A のリソースの作成またはアクセスが許可されます。場合によっては、AWS のサービスにロールを引き受けるためのアクセス許可を付与する必要があります。このアプローチをサポートするために、信頼ポリシーのプリンシパルを AWS のサービスのプリンシパルにすることもできます。

IAM を使用した許可の委任の詳細については、「IAM ユーザーガイド」の「アクセス管理」を参照してください。

以下に示しているのは、お客様の AWS アカウントに対する DescribeCacheClusters アクションの実行をユーザーに許可するポリシーの例です。ElastiCache では、API アクションのリソース ARN を使用した特定のリソースの識別もサポートしています。(このアプローチは、リソースレベルのアクセス許可とも呼ばれます)。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "DescribeCacheClusters",
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

ElastiCache でアイデンティティベースのポリシーを使用する場合の詳細については、「Amazon ElastiCache でのアイデンティティベースのポリシー (IAM ポリシー) の使用」を参照してください。ユーザー、グループ、ロール、アクセス許可の詳細については、IAM ユーザーガイドの「アイデンティティ (ユーザー、グループ、ロール)」を参照してください。

ポリシー要素の指定: アクション、効果、リソース、プリンシパル

サービスは、Amazon ElastiCache リソースごとに (Amazon ElastiCache のリソースとオペレーション を参照)、一連の API オペレーションを定義します (アクションを参照)。こうした API オペレーションへのアクセス権限を付与するために、ElastiCache は一連のアクションをポリシーに定義します。例えば、ElastiCache クラスターリソースに対して、アクション CreateCacheCluster、DeleteCacheCluster、DescribeCacheCluster を定義します。1 つの API オペレーションの実行で、複数のアクションのアクセス権限が必要になる場合があります。

最も基本的なポリシーの要素を次に示します。

- リソース– ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「Amazon ElastiCache のリソースとオペレーション」を参照してください。
- アクション - アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。たとえば、指定した `Effect` に応じて、`elasticache:CreateCacheCluster` アクセス許可では、Amazon ElastiCache `CreateCacheCluster` オペレーションの実行をユーザーに許可または拒否します。
- 効果 – ユーザーが特定のアクションを要求する際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に付与 (許可) していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。たとえば、別のポリシーでリソースへのアクセスが許可されているユーザーに対して、そのリソースへのアクセスを禁止できます。
- プリンシパル - ID ベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

IAM ポリシーの構文と記述の詳細については、「IAM ユーザーガイド」の「AWS IAM ポリシーリファレンス」を参照してください。

すべての Amazon ElastiCache API アクションを示す表については、「ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス」を参照してください。

ポリシーでの条件の指定

許可を付与するとき、IAM ポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。例えば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリシー言語での条件の指定の詳細については、「IAM ユーザーガイド」の「条件」を参照してください。

条件を表すには、あらかじめ定義された条件キーを使用します。ElastiCache 固有の条件キーを使用するには、「条件キーの使用」を参照してください。必要に応じて使用できる AWS 全体の条件キーがあります。AWS 全体を対象とするすべてのキーのリストについては、「IAM ユーザーガイド」の「条件に利用可能なキー」を参照してください。

Amazon ElastiCache 用の AWS マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースで権限を提供できるように設計されているため、ユーザー、グループ、ロールへの権限の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権の権限を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別にカスタマー管理ポリシーを定義することで、権限を絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されている権限を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS サービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: ElastiCacheServiceRolePolicy

IAM エンティティに ElastiCacheServiceRolePolicy をアタッチすることはできません。このポリシーは、ユーザーに代わって ElastiCache がアクションを実行することを許可する、サービスリンクロールにアタッチされます。

このポリシー は、ElastiCache がユーザーの代わってキャッシュを管理するため、必要に応じて AWS リソースを管理できるものです。

- `ec2` — VPC エンドポイント (サーバーレスキャッシュ用)、Elastic Network Interface (ENI) (独自設計型クラスター用)、セキュリティグループなど、キャッシュノードに接続する EC2 ネットワークリソースを管理します。
- `cloudwatch` — サービスから CloudWatch にメトリクスデータを送信します。
- `outposts` — AWS Outpost でのキャッシュノードの作成を許可します。

ElastiCacheServiceRolePolicy は IAM コンソールに、ElastiCacheServiceRolePolicy は「AWS マネージドポリシーリファレンスガイド」に記載されています。

AWS マネージドポリシー: AmazonElastiCacheFullAccess

`AmazonElastiCacheFullAccess` ポリシーは IAM ID に添付できます。

このポリシーにより、プリンシパルは AWS マネジメントコンソールを使用して ElastiCache にフルアクセスできます。

- `elasticache` — すべての API にアクセスできます。
- `iam` — サービスの運用に必要なサービスリンクロールを作成します。
- `ec2` — キャッシュ作成に必要な依存型 EC2 リソース (VPC、サブネット、セキュリティグループ) を記述し、VPC エンドポイント (サーバーレスキャッシュ用) の作成を許可します。
- `kms` — カスタマー管理の CMK を保管時の暗号化に使用できるようにします。
- `cloudwatch` — コンソールに ElastiCache メトリクスを表示するためのメトリクスへのアクセスを許可します。
- `application-autoscaling` — キャッシュの自動スケーリングポリシーを記述するためのアクセスを許可します。
- `logs` — コンソールのログ配信機能のログストリームを入力するために使用されます。
- `firehose` — コンソールのログ配信機能の配信ストリームを入力するために使用されます。
- `s3` — コンソールのスナップショット復元機能用の S3 バケットを入力するために使用されます。
- `outposts` — コンソールでキャッシュを作成するための AWS Outpost を入力するために使用されます。
- `sns` — コンソールの通知機能用の SNS トピックを入力するために使用されます。

AmazonElastiCacheFullAccess ポリシーは IAM コンソールに、AmazonElastiCacheFullAccess は「AWS マネージドポリシーリファレンスガイド」に記載されています。

AWS マネージドポリシー: AmazonElastiCacheReadOnlyAccess

`AmazonElastiCacheReadOnlyAccess` ポリシーは IAM ID に添付できます。

このポリシーは、プリンシパルが AWS マネジメントコンソールを使用して ElastiCache に読み取り専用でアクセスすることを許可します。

- `elasticache` — 読み取り専用 `Describe` API にアクセスできます。

AmazonElastiCacheReadOnlyAccess ポリシーは IAM コンソール
に、AmazonElastiCacheReadOnlyAccess は「AWS マネージドポリシーリファレンスガイド」に記載されています。

ElastiCache 用の AWS マネージドポリシーの更新

ElastiCache 用の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページの変更に関する自動通知については、ElastiCache ドキュメントの履歴ページの RSS フィードをサブスクライブしてください。

変更	説明	日付
AmazonElastiCacheFullAccess — 既存のポリシーの更新	ElastiCache に、サーバーレスキャッシュを管理し、コンソールからすべてのサービス機能を使用できるようにする新しいアクセス許可が追加されました。	2023 年 11 月 27 日
ElastiCacheServiceRolePolicy — 既存のポリシーの更新	ElastiCache に、サーバーレスキャッシュリソースの VPC エンドポイントを管理するための新しいアクセス許可が追加されました。	2023 年 11 月 27 日
ElastiCache が変更の追跡を開始	ElastiCache 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 ElastiCache でのアイデンティティベースのポリシー (IAM ポリシー) の使用

このトピックでは、アカウント管理者が IAM ID (ユーザー、グループ、ロール) へのアクセス許可ポリシーをアタッチする、ID ベースのポリシーの例を示します。

> **Important**
>
> 最初に、Amazon ElastiCache リソースへのアクセスを管理するための基本的な概念とオプションについて説明するトピックを読むことをお勧めします。詳細については、「ElastiCache リソースに対するアクセス許可の管理の概要」を参照してください。

このセクションでは、次のトピックを対象としています。

- Amazon ElastiCache 用の AWS マネージドポリシー
- カスタマーマネージドポリシーの例

以下に示しているのは、アクセス許可ポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowClusterPermissions",
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache",
            "elasticache:CreateCacheCluster",
            "elasticache:DescribeServerlessCaches",
            "elasticache: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 アクションのアクセス許可を付与します (`elasticache:Create*`、`elasticache:Describe*`、`elasticache:Modify*`)
- 2 番目のステートメントは、`Resource` 値の最後に指定した IAM ロール名での IAM アクション`iam:PassRole`のアクセス許可を付与します。

ID ベースのポリシーでアクセス許可を得るプリンシパルを指定していないため、ポリシーでは `Principal` 要素を指定していません。ユーザーにポリシーをアタッチすると、そのユーザーが暗黙のプリンシパルになります。IAM ロールにアクセス権限ポリシーをアタッチすると、ロールの信頼ポリシーで識別されたプリンシパルがアクセス権限を得ることになります。

すべての Amazon ElastiCache API アクションとそれらが適用されるリソースの表については、「ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス」を参照してください。

カスタマーマネージドポリシーの例

デフォルトポリシーを使用せず、カスタム管理ポリシーを使用することを選択した場合は、以下の 2 点のいずれかを確認してください。`iam:createServiceLinkedRole` を呼び出すためのアクセス許可があることが必要です (詳細については、「例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する」を参照)。または、ElastiCache サービスにリンクされたロールを作成済みであることが必要です。

Amazon ElastiCache コンソールを使用するために必要な最小限のアクセス権限と組み合わせて、このセクションでのポリシーの例は、追加のアクセス権限を付与します。この例は、AWS SDK と AWS CLI に関連しています。

IAM ユーザーおよびグループのセットアップ手順については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

> **Important**
>
> IAM ポリシーは必ず、本稼働環境での使用前にテストしてください。ElastiCache のアクションによっては、シンプルに見えても、ElastiCache コンソールの使用時にそれらのアクションをサポートするために、他のアクションが必要になる場合があります。たとえば、`elasticache:CreateCacheCluster` は、ElastiCache キャッシュクラスターを作成するためのアクセス許可を付与します。ただし、このオペレーションを実行するために、ElastiCache コンソールでは `Describe` と `List` の多数のアクションが使用されて、リストが事前設定されます。

例

- 例 1: ユーザーに ElastiCache リソースへの読み取り専用アクセスを許可する
- 例 2: ユーザーに一般的な ElastiCache システム管理者タスクの実行を許可する
- 例 3: ユーザーにすべての ElastiCache API アクションへのアクセスを許可する
- 例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する

例 1: ユーザーに ElastiCache リソースへの読み取り専用アクセスを許可する

以下のポリシーでは、リソースを一覧表示する ElastiCache アクションを実行するためのアクセス許可をユーザーに付与します。通常、このタイプのアクセス権限ポリシーは管理者グループにアタッチします。

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECReadOnly",
      "Effect":"Allow",
      "Action": [
          "elasticache:Describe*",
          "elasticache:List*"],
      "Resource":"*"
      }
   ]
}
```

例 2: ユーザーに一般的な ElastiCache システム管理者タスクの実行を許可する

一般的なシステム管理者タスクには、リソースの変更が含まれます。　システム管理者は ElastiCache イベントに関する情報を取得することが必要になる場合もあります。以下のポリシーでは、これらの一般的なシステム管理タスクに必要な ElastiCache アクションを実行するためのアクセス権限をユーザーに付与します。通常、このタイプのアクセス権限ポリシーはシステム管理者グループにアタッチします。

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "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

以下のポリシーでは、ユーザーにすべての ElastiCache アクションへのアクセスを許可します。このタイプのアクセス権限ポリシーは管理者ユーザーにのみ付与することをお勧めします。

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowAll",
      "Effect":"Allow",
      "Action":[
          "elasticache:*"
      ],
      "Resource":"*"
      }
   ]
}
```

例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する

次のポリシーでは、ユーザーがIAM `CreateServiceLinkedRole` API を呼び出すことを許可します。mutative ElastiCache オペレーションを実行するユーザーには、このタイプのアクセス許可ポリシーを与えることをお勧めします。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
```

```
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWSServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

リソースレベルのアクセス許可

IAM ポリシーでリソースを指定することで、アクセス許可の範囲を制限できます。ElastiCache API アクションの多くは、アクションの動作に応じて異なるリソースタイプをサポートしています。各 IAM ポリシーステートメントによって、リソースで実行されるアクションに対するアクセス許可が付与されます。アクションが名前の付いたリソースで動作しない場合、またはすべてのリソースに対してアクションを実行するアクセス許可を付与した場合、ポリシー内のリソースの値はワイルドカード (*) になります。多くの API アクションでは、リソースの Amazon リソースネーム (ARN)、または複数のリソースに一致する ARN パターンを指定することによって、ユーザーが変更できるリソースを制限できます。リソース別にアクセス許可を制限するには、ARN 別にリソースを指定します。

ElastiCache リソースのタイプとその ARN のリストを確認するには、「サービス認証リファレンス」の「Amazon ElastiCache で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon ElastiCache で定義されるアクションを参照してください。

例

- 例 1: 特定の ElastiCache リソースタイプへのフルアクセスをユーザーに許可する
- 例 2: サーバーレスキャッシュへのユーザーアクセスを拒否する。

例 1: 特定の ElastiCache リソースタイプへのフルアクセスをユーザーに許可する

次のポリシーでは、すべてのリソースタイプのサーバーレスキャッシュを明示的に許可します。

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "elasticache:*",
        "Resource": [
             "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
```

```
        ]
}
```

例 2: サーバーレスキャッシュへのユーザーアクセスを拒否する。

次の例では、特定のサーバーレスキャッシュへのアクセスを明示的に拒否します。

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "elasticache:*",
        "Resource": [
            "arn:aws:elasticache:us-east-1:account-id:serverlesscache:name"
        ]
}
```

条件キーの使用

IAM ポリシーを有効にする方法を決める条件を指定できます。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のキーをポリシーで指定したキー値と比較できます。ポリシー要素の詳細については、IAM JSON policy elements: Condition を参照してください。

ElastiCache 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、特定のリクエストに対して個別のパラメータセットを制御するための条件を指定した IAM アクセス許可ポリシーを作成します。次に、IAM コンソールを使用して作成する IAM ユーザー、グループ、またはロールにそのポリシーを適用します。

条件を適用するには、条件情報を IAM ポリシーステートメントに追加します。次の例では、独自に設計されたすべてのキャッシュクラスターがノードタイプ cache.r5.large になるという条件を指定します。

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "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、「タグベースのアクセスコントロールポリシーの例」を参照してください。

ポリシー条件演算子の使用に関する詳細については、「ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス」を参照してください。

ポリシー例: きめ細かなパラーメータコントロールのための IAM ポリシー条件の使用

このセクションでは、前述の 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:MaximumECPUPerSecond : サーバーレスキャッシュの 1 秒あたりの最大 ECPU 値を指定します。指定された条件では、1 秒あたりに特定の数を超える ECPU を実行できるキャッシュを作成することはできません。

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

```
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:MaximumECPUPerSecond": "100000"
                }
            }
        }
    ]
}
```

3. elasticache:CacheNodeType: ユーザーが作成できる NodeType(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"
                    ]
                }
            }
        }
    ]
}
```

4. elasticache:EngineVersion: 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"
                }
            }
        }
    ]
}
```

5. 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"
            }
        }
    }
  ]
```

```
}
```

6. elasticache:CacheParameterGroupName: クラスター上の組織からの特定のパラメータを使用して、デフォルト以外のパラメータグループを指定します。パラメータグループの命名パターンを指定したり、特定のパラメータグループ名に対するブロック削除を指定することもできます。以下は、「」のみの使用を制限する例ですmy-org-param-group。

```
{
    "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"
                }
            }
        }
    ]
}
```

7. elasticache: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"
                    ]
                }
            }
```

```
        }
    ]
}
```

8. elasticache:CacheNodeType: cacheNodeType cache.r5.large または cache.r6g.4xlarge CreateCacheClusterで を許可し、 にタグを付けます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 ElastiCache でのサービスにリンクされたロールの使用

Amazon ElastiCache は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用しています。サービスにリンクされたロールは、Amazon ElastiCache などの AWS サービスに直接リンクされた一意のタイプの IAM ロールです。Amazon ElastiCache サービスにリンクされたロールは、Amazon ElastiCache によって事前定義されています。それらには、サービスがユーザーのクラスターに代わって AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれます。

サービスにリンクされたロールを使用することで、必要なアクセス権限を手動で追加する必要がなくなるため、Amazon ElastiCache の設定が簡単になります。ロールは AWS アカウント内にありますが、Amazon ElastiCache のユースケースにリンクされており、アクセス権限が事前に定義されています。これらのロールを引き受けることができるのは Amazon ElastiCache のみで、事前定義されたアクセス許可ポリシーを使用することができるのはこれらのロールのみです。ロールを削除するには、まず関連リソースを削除します。これにより、リソースにアクセスするのに必要なアクセス許可を誤って削除することがなくなり、Amazon ElastiCache リソースは保護されます。

サービスリンクロールをサポートする他のサービスについては、IAM と連携する AWS のサービスを参照して、[Service-linked role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを見つけてください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

目次

- Amazon ElastiCache のサービスにリンクされたロールの説明の編集
  - サービスにリンクされたロールの説明の編集 (IAMコンソール)
  - サービスにリンクされたロールの説明の編集 (IAM CLI)
  - サービスにリンクされたロールの説明の編集 (IAM API)
- Amazon ElastiCache でのサービスにリンクされたロールの削除
  - サービスにリンクされたロールのクリーンアップ
  - サービスにリンクされたロールの削除 (IAMコンソール)
  - サービスにリンクされたロールの削除 (IAM CLI)
  - サービスにリンクされたロールの削除 (IAM API)

Amazon ElastiCache でのサービスにリンクされたロールのアクセス許可

サービスリンクロールを作成するためのアクセス許可

IAM エンティティが AWS ServiceRoleForElastiCache サービスリンクロールを作成することを許可するには

以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

IAM エンティティが AWS ServiceRoleForElastiCache サービスリンクロールを削除することを許可するには

以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

または AWS 管理ポリシーを使用して、Amazon ElastiCache へのフルアクセスを許可することもできます。

サービスにリンクされたロールの作成 (IAM)

IAM コンソール、CLI または API を使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールの作成 (IAM コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。次に、新しいロールの作成を選択します。
3. [Select type of trusted entity] (信頼されたエンティティの種類を選択) の下で、[AWS Service] (サービス) を選択します。
4. [Or select a service to view its use cases] (またはサービスを選択してそのユースケースを表示) で、[ElastiCache] を選択します。
5. [Next: Permissions] (次のステップ: アクセス許可) を選択します。
6. ポリシー名 の下で、`ElastiCacheServiceRolePolicy` はこのロールに必要であることに注意してください。次: タグ を選択します。
7. タグは、サービスにリンクされたロールではサポートされないことに注意してください。次: レビュー を選択します。
8. 「オプショナル」ロールの説明 で、サービスにリンクされた新しいロールの説明を編集します。
9. ロール情報を確認し、ロールの作成 を選択します。

サービスにリンクされたロールの作成 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (CLI)

次のオペレーションを使用してください。

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

サービスにリンクされたロールの作成 (IAM API)

IAM API を使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (API)

CreateServiceLinkedRole API コールを使用します。リクエストで、サービス名`elasticache.amazonaws.com`を指定します。

Amazon ElastiCache のサービスにリンクされたロールの説明の編集

Amazon ElastiCache では、AWSServiceRoleForElastiCache のサービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。

サービスにリンクされたロールの説明の編集 (IAMコンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

サービスにリンクされたロールの説明を編集するには (コンソール)

1. IAM コンソールのナビゲーションペインで [ロール] を選択します。
2. 変更するロールの名前を選択します。
3. ロールの説明の右端にある編集を選択します。
4. ボックスに新しい説明を入力し、保存を選択します。

サービスにリンクされたロールの説明の編集 (IAM CLI)

サービスにリンクされたロールの説明は、AWS Command Line Interface から IAM オペレーションを使用して編集できます。

サービスにリンクされたロールの説明を変更するには (CLI)

1. 「オプショナル」現在のロールの説明を表示するには、IAM オペレーション `get-role` の AWS CLI を使用します。

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForElastiCache
   ```

   CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。たとえば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。
2. サービスにリンクされたロールの説明を更新するには、IAM オペレーション `update-role-description` の AWS CLI を使用します。

   Linux、macOS、Unix の場合:

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForElastiCache \
       --description "new description"
   ```

   Windows の場合:

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForElastiCache ^
       --description "new description"
   ```

サービスにリンクされたロールの説明の編集 (IAM API)

サービスにリンクされたロールの説明は、IAM API を使用して編集できます。

サービスにリンクされたロールの説明を変更するには (API)

1. 「オプショナル」現在のロールの説明を表示するには、IAM API オペレーション GetRole を使用します。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. ロールの説明を更新するには、IAM API オペレーション UpdateRoleDescription を使用します。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &Description="New description"
   ```

Amazon ElastiCache でのサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニターリングされたり、メンテナンスされたりすることがなくなります。ただし、削除する前に、サービスにリンクされた役割をクリーンアップする必要があります。

Amazon ElastiCache はサービスにリンクされたロールを削除しません。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそれに関連付けられているリソース、クラスターがないことを確認する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。AWSServiceRoleForElastiCache ロールのチェックボックスではなく、ロールの名前を選択します。
3. 選択したロールの 概要 ページで、アクセスアドバイザー タブを選択します。
4. アクセスアドバイザー タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

AWSServiceRoleForElastiCache が必要な Amazon ElastiCache リソースを削除するには (コンソール)

- クラスターを削除するには、以下を参照してください。
  - AWS Management Consoleの使用
  - AWS CLIの使用
  - ElastiCache API の使用

サービスにリンクされたロールの削除 (IAMコンソール)

IAM コンソールを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。ロール名または行そのものではなく、削除するロール名の横にあるチェックボックスをオンにします。
3. ページ上部にある [ロールのアクション] で [ロールの削除] を選択します。
4. 確認ダイアログボックスで、サービスの最終アクセス時間データを確認します。これは、選択したそれぞれのロールの AWS サービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。

5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送信すると、削除タスクは成功または失敗する可能性があります。タスクが失敗した場合は、通知から 詳細を表示または リソースを表示を選択して、削除が失敗した理由を知ることができます。

サービスにリンクされたロールの削除 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (CLI)

1. 削除するサービスにリンクされたロールの名前が分からない場合、以下のコマンドを入力します。このコマンドでは、アカウントにあるロールとその Amazon リソースネーム (ARN) を一覧表示します。

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールに ARN `arn:aws:iam::123456789012:role/myrole` がある場合、そのロールを **`myrole`** と参照します。
2. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `deletion-task-id` を取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、以下を入力します。

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```
3. 削除タスクのステータスを確認するには、以下を入力します。

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   削除タスクのステータスは、 `NOT_STARTED`、`IN_PROGRESS`, `SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

サービスにリンクされたロールの削除 (IAM API)

IAM API を使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (API)

1. サービスにリンクされたロールの削除リクエストを送信するには、DeleteServiceLinkedRole を呼び出します。リクエストで、ロール名を指定します。

   サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `DeletionTaskId` を取得して、削除タスクのステータスを確認する必要があります。
2. 削除タスクのステータスを確認するには、GetServiceLinkedRoleDeletionStatus を呼び出します。リクエストで `DeletionTaskId` を指定します。

   削除タスクのステータスは、 `NOT_STARTED`、`IN_PROGRESS`, `SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

ElastiCache API アクセス許可: アクション、リソース、および条件リファレンス

アクセスコントロール を設定し、IAM ポリシーにアタッチするアクセス許可ポリシー (アイデンティティベースまたはリソースベース) を作成するときは、以下の表をリファレンスとして使用できます。この表には、各 Amazon ElastiCache API オペレーションと、アクションを実行するためのアクセス許可を付与できる対応するアクションが一覧表示されています。ポリシーの `Action` フィールドでアクションを指定し、ポリシーの `Resource` フィールドでリソースの値を指定します。特に明記されていない限り、リソースは必須です。一部のフィールドには、必須リソースとオプションリソースの両方が含まれます。リソース ARN がない場合、ポリシー内のリソースはワイルドカード (*) になります。

ElastiCache ポリシーで条件キーを使用して条件を表現できます。 ElastiCache固有の条件キーのリストと、それらが適用されるアクションとリソースタイプを確認するには、「」を参照してください条件キーの使用。 AWS全体のキーの完全なリストについては、「IAM ユーザーガイド」のAWS「 グローバル条件コンテキストキー」を参照してください。

> **Note**
>
> アクションを指定するには、API オペレーション名 (`elasticache:DescribeCacheClusters` など) の前に `elasticache:` プレフィックスを使用します。

ElastiCache アクションのリストを確認するには、「サービス認証リファレンス」の「Amazon で定義されるアクション ElastiCache」を参照してください。

Amazon のコンプライアンス検証 ElastiCache

サードパーティーの監査者は、、SOC、PCIFed 、 など、複数の コンプライアンスプログラムの一環として AWS サービスのセキュリティRAMPと AWS コンプライアンスを評価しますHIPAA。

AWS サービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS サービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS サービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービス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対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべての AWS 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、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド — コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS 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。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS サービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS サービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことでDSS、 PCI などのさまざまなコンプライアンス要件への対応に役立ちます。

- AWS Audit Manager – これにより AWS サービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

詳細情報

AWS クラウドコンプライアンスの一般的な情報については、以下を参照してください。

- FIPS サービス別のエンドポイント
- でのサービスの更新 ElastiCache
- AWS クラウドコンプライアンス
- 責任共有モデル
- AWS PCI DSS コンプライアンスプログラム

Amazon ElastiCache s の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon ElastiCache では、AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズに対応できるように複数の機能を提供しています。

トピック

- 障害の軽減

障害の軽減

Amazon ElastiCache の実装を計画するときは、障害がアプリケーションやデータに与える影響を最小限に抑えるように計画する必要があります。このセクションのトピックでは、アプリケーションおよびデータを障害から保護するために実行できるアプローチについて説明します。

トピック

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、1 ~ 60 個のノード、または特別なリクエストで作成できます。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 クラスターに 60 個を超えるノードが必要な場合、または AWS リージョンに合計 300 個を超えるノードが必要な場合は、https://aws.amazon.com/contact-us/elasticache-node-limit-request/ の ElastiCache 「制限引き上げリクエスト」フォームに入力します。

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。AZ の障害が発生した場合には、他の AZ でキャッシュされたデータではなく、その AZ でキャッシュされたデータが失われます。

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。

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

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 AZ 間に均等に分散することで、予期しない AZ の障害が発生した場合の影響を最小化します。これはサーバーレスキャッシュでは自動的に行われます。

AWS ElastiCache でのインフラストラクチャセキュリティ

マネージドサービスとして、AWS ElastiCache は、AWS アーキテクチャセンターの「セキュリティとコンプライアンスセクション」で説明されている AWS グローバルネットワークセキュリティ手順によって保護されます。

AWS が公開している API コールを使用して、ネットワーク経由で ElastiCache にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。TLS 1.3 以降が推奨されます。また、一時的ディフィー・ヘルマン Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS STS を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

でのサービスの更新 ElastiCache

ElastiCache は、キャッシュ、クラスター、ノードのフリートを自動的にモニタリングして、サービスの更新が利用可能になったときにそれらを適用します。サーバーレスキャッシュのサービス更新は自動的かつ透過的に適用されます。自己設計クラスターの場合、 がこれらの更新 ElastiCache を適用できるように、事前定義されたメンテナンスウィンドウを設定します。ただし、場合によっては、このアプローチが厳格すぎて、ビジネスフローが制限される可能性もあります。

サービス更新では、独自設計型クラスターにアップデートを適用するタイミングと内容を制御できます。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。

サービス更新の管理

ElastiCache 独自設計型クラスターの 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独自設計型クラスターが 1 つ以上ある場合は、更新がリリースされると、E メール、、Personal Health Dashboard (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 コンソールのサービス更新ページにも表示されます。このダッシュボードを使用すると、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を適用できるように、セキュリティ更新タイプの更新プログラム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。

トピック

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は、利用可能になったときにアクションを実行しないように選択できます。 ElastiCache は、自動更新の開始日 より後に、クラスターの今後のメンテナンスウィンドウ中に更新を適用するようにスケジュールします。更新のステージごとに、関連する通知を受け取ります。

> **Note**
>
> ステータスが 使用可能または スケジュール済み であるサービスの更新だけを適用できます。

該当する 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になったら、 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。
   - 日付による推奨適用 ： ElastiCache 更新を適用するガイダンスの日付。
   - [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
  ```

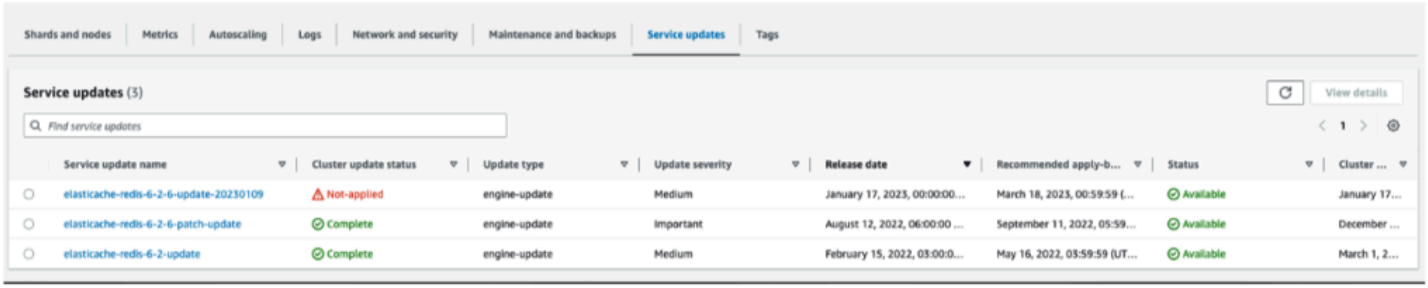
詳細については、「」を参照してくださいbatch-apply-update-action。

AWS コンソールを使用して最新のサービス更新が適用されていることを確認する

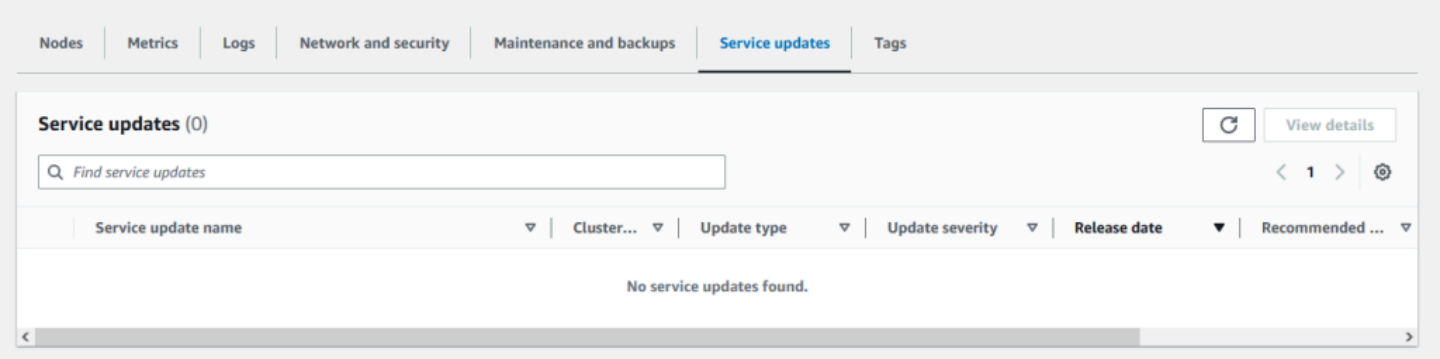
ElastiCache (Redis OSS) クラスターが最新のサービス更新を実行していることを確認するには、次の手順に従います。

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 を選択します。



コンソールに「サービスの更新が見つかりません」と表示される場合は、 ElastiCache (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 コンソールの ElastiCache ダッシュボードの上部に更新の表示/停止タブが表示されます。
- 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。

Amazon ElastiCache でのログ記録とモニタリング

キャッシュを管理するには、キャッシュのパフォーマンスを把握することが重要です。ElastiCache は、キャッシュのパフォーマンスをモニタリングするために Amazon CloudWatch Logs へ発行されるメトリクスを生成します。さらに、ElastiCache は、キャッシュリソースに重大な変更 (新しいキャッシュの作成やキャッシュの削除など) が発生したときにイベントを生成します。

トピック

- AWS CloudTrail を使用した Amazon ElastiCache API コール

サーバーレスのメトリクスとイベント

このセクションでは、サーバーレスキャッシュを使用する際にモニタリングできるメトリクスとイベントを説明しています。

トピック

サーバーレスキャッシュメトリクス

`AWS/ElastiCache` 名前空間には、Memcached サーバーレスキャッシュの次の CloudWatch メトリクスが含まれます。

メトリクス	説明	単位
BytesUsedForCache	キャッシュに保存されているデータによって使用される総バイト数。	バイト

メトリクス	説明	単位
ElastiCacheProcessingUnits	キャッシュ上で実行されたリクエストによって消費された ElastiCacheProcessingUnits (ECPU) の総数。	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
CurrConnections	キャッシュへのクライアント接続の数。	カウント
ThrottledCmds	ワークロードが ElastiCache のスケーリングよりも速くスケーリングしたため、ElastiCache によってスロットリングされたリクエストの数。	カウント
NewConnections	この期間内にサーバーによって受け入れられた接続の総数。	カウント
CurrItems	キャッシュの項目数。	カウント
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

メトリクス	説明	単位
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、以下の 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
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

メトリクス	説明	単位
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
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

メトリクス	説明	単位
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記録します。この情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などがあります。ElastiCache コンソール、AWS CLI describe-events コマンド、または ElastiCache API アクション `DescribeEvents` を使用して、ログから簡単にイベントを取得できます。

Amazon EventBridge を使用して ElastiCache イベントのモニタリング、インジェスト、変換、処理できます。詳細については、Amazon EventBridge https://docs.aws.amazon.com/eventbridge/latest/userguide/ を参照してください。

ElastiCache イベントの表示 (コンソール)

ElastiCache コンソールを使用してスタックイベントを表示するには:

1. AWS Management Consoleにサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 利用可能なすべてのイベントのリストを表示するには、ナビゲーションペインで [Events] を選択します。
3. [イベント] 画面では、リストの各行が 1 つのイベントを表し、イベントソース、イベントタイプ、イベントの GMT 時間、イベントの説明が表示されます。Filter を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

ElastiCache イベントの表示 (AWS CLI)

AWS CLI を使用して ElastiCache イベントのリストを作成するには、describe-events コマンドを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、最大 40 個のサーバーレスキャッシュイベントを一覧表示します。

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

次のコードでは、過去 24 時間 (1440 分) のサーバーレスキャッシュのイベントをすべて一覧表示します。

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

サーバーレスイベント

このセクションでは、サーバーレスキャッシュに対して受け取る可能性のあるさまざまなタイプのイベントについて説明します。

サーバーレスキャッシュ作成イベント

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ作成	キャッシュ ARN	作成	サーバーレスキャッシュ	キャッシュ <cache-name> が作成され、使用できる状態になりました。
キャッシュ作成失敗	キャッシュ ARN	失敗	サーバーレスキャッシュ	キャッシュ <cache-name> を作成できませんでした。VPC エンドポイントを作成するための空き IP アドレスが不足しています。

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。リクエストで入力されたサブネットが無効です。
キャッシュ作成失敗	キャッシュ ARN	失敗	サーバーレスキャッシュ	キャッシュ <cache-name> を作成できませんでした。VPC エンドポイントの作成のクォータ上限に達しました。
キャッシュ作成失敗	キャッシュ ARN	失敗	サーバーレスキャッシュ	キャッシュ <cache-name> を作成できませんでした。VPC エンドポイントを作成するアクセス許可がありません。

サーバーレスキャッシュ更新イベント

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> 用に SecurityG

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
				roups が更新されました。
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> のタグが更新されました。
キャッシュ更新失敗	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> の更新に失敗しました。SecurityGroups 更新に失敗しました。
キャッシュ更新失敗	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> の更新に失敗しました。SecurityGroups 更新は、アクセス許可が不十分なため、失敗しました。
キャッシュ更新失敗	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> の更新に失敗しました。SecurityGroups が無効なため、SecurityGroups 更新に失敗しました。

サーバーレスキャッシュ削除イベント

詳細タイプ	リソースリスト	カテゴリ	ソース	メッセージ
キャッシュ削除	キャッシュ ARN	削除	サーバーレスキャッシュ	キャッシュ <cache-name> が削除されました。

サーバーレスキャッシュ使用量上限イベント

詳細タイプ	説明	単位	ソース	メッセージ
キャッシュ更新	キャッシュ ARN	設定変更	サーバーレスキャッシュ	キャッシュ <cache-name> の上限が更新されました。
キャッシュ更新失敗	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。

独自設計型のクラスター、メトリクス、イベント

このセクションでは、独自設計型クラスターを使用する際に予想されるメトリクス、イベント、ログについて説明します。

トピック

独自設計型クラスターのメトリクス

クラスターを独自に設計すると、ElastiCache はホストレベルのメトリクスとキャッシュメトリクスの両方を含むメトリクスを各ノードレベルで出力します。

Memcached の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。

Memcached のノードレベルのメトリクスの詳細については、「Memcached のメトリクス」を参照してください。

独自設計型クラスターのイベント

ElastiCache は、独自設計型キャッシュに関連するイベントを記録します。独自設計型クラスターを使用する場合は、 ElastiCache コンソール、、または Amazon Simple Notification Service (SNS) を使用して AWS CLIクラスターイベントを表示できます。独自設計型クラスターイベントは Amazon には公開されません 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、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。
3. [イベント] 画面のリスト内の各行は 1 つのイベントを表し、イベントの送信元、イベントのタイプ、イベントの時間 (GMT)、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、 コマンド describe-events 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 Virtual Private Cloud (VPC) に新しいキャッシュクラスターに起動する試みが行われました。
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : *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。 • 1 つ以上の新しい EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループで承認されました。 • 1 つ以上の EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループから取り消されました。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストのパフォーマンスが低下している、または到達できないことを検出し、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 ElastiCache は、 がキャッシュノードを置き換えた後でもキャッシュノードの使用を停止することがあります。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パフォーマンスが低下している、または到達できないことを検出し、キャッシュノードの置き換えを完了しました。 **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 ElastiCache は、 がキャッシュノードを置き換えた後でもキャッシュノードの使用を停止することがあります。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 (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 (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。 実行できるアクションについては、「Memcached のキャッシュノードの交換」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。 実行できるアクションについては、「Memcached のキャッシュノードの交換」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 (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 (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。

CloudWatch メトリクスの使用状況のモニタリング

ElastiCache には、クラスターを監視できるようにするメトリクスが用意されています。CloudWatch を通じてこれらのメトリクスにアクセスできます。CloudWatch の詳細については、「CloudWatch のドキュメント」を参照してください。

ElastiCache では、ホストレベルのメトリクス (たとえば、CPU 使用率など) とキャッシュエンジンソフトウェアに固有のメトリクス (たとえば、キャッシュの取得やキャッシュの損失など) の両方が提供されます。これらのメトリクスは 60 秒間隔で各キャッシュノードに対して測定およびパブリッシュされます。

> **Important**
>
> キャッシュクラスターのパフォーマンスが低下し始めた場合に通知を受け取ることができるように、特定の主要メトリクスに CloudWatch アラームを設定することを検討してください。詳細については、このガイドの「モニタリングすべきメトリクス」を参照してください。

トピック

ホストレベルのメトリクス

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。

以下の資料も参照してください。

- Memcached のメトリクス

メトリクス	説明	単位
CPUUtilization	ホスト全体の CPU 使用率の割合 (%)。	割合 (%)
CPUCreditBalance	インスタンスが起動または開始後に蓄積した獲得 CPU クレジットの数。T2 スタンダードの場合、CPUCreditBalance には蓄積された起動クレジットの数も含まれます。	クレジット (vCPU 分)

メトリクス	説明	単位
	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、起動クレジットは制限に対してカウントされません。 CPUCreditBalance のクレジットは、インスタンスがそのベースライン CPU 使用率を超えてバーストするために消費できます。 CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。	
`CPUCreditUsage`	CPU 使用率に関してインスタンスで消費される CPU クレジットの数。1 つの CPU クレジットは、1 個の vCPU が 100% の使用率で 1 分間実行されること、または、vCPU、使用率、時間の同等の組み合わせ (例えば、1 個の vCPU が 50% の使用率で 2 分間実行されるか、2 個の vCPU が 25% の使用率で 2 分間実行される) に相当します。 CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。5 分を超える期間を指定する場合は、Average 統計の代わりに Sum 統計を使用します。	クレジット (vCPU 分)
`FreeableMemory`	ホストで使用可能な空きメモリの量。OS によって解放できる可能性があるとレポートされる RAM、バッファ、およびキャッシュから算出されます。	バイト

メトリクス	説明	単位
NetworkBytesIn	ホストがネットワークから読み取ったバイト数。	バイト
NetworkBytesOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたバイトの数。	バイト
NetworkPacketsIn	すべてのネットワークインターフェイスでの、このインスタンスによって受信されたパケットの数。このメトリクスは、受信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkPacketsOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたパケットの数。このメトリクスは、送信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkBandwidthInAllowanceExceeded	インバウンドの集計帯域幅がインスタンスの最大値を超えたために形成されたパケットの数。	カウント
NetworkConntrackAllowanceExceeded	接続トラッキングがインスタンスの最大数を超え、新しい接続を確立できなかったために形成されたパケットの数。これにより、インスタンスとの間で送受信されるトラフィックのパケット損失が発生する可能性があります。	カウント
NetworkBandwidthOutAllowanceExceeded	アウトバウンド集計帯域幅がインスタンスの最大値を超えたために形成されたパケットの数。	カウント
Network Packets Per Second Allowance Exceeded	1 秒あたりの双方向パケットがインスタンスの最大値を超えたために形成されたパケットの数。	カウント
NetworkMaxBytesIn	1 分あたりの受信バイトの最大バースト。	バイト
NetworkMaxBytesOut	1 分あたりの送信バイトの最大バースト。	バイト

メトリクス	説明	単位
NetworkMaxPacketsIn	1 分あたりの受信パケットの最大バースト。	カウント
NetworkMaxPacketsOut	1 分あたりの送信パケットの最大バースト。	カウント
SwapUsage	ホストで使用されるスワップの量。	バイト

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 名前空間には、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	キャッシュが受信したが、その Cas (チェックと設定) 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。	Count (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。	Count (カウント)

メトリクス	説明	単位
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。	Count (カウント)
CmdFlush	キャッシュが受信した flush コマンドの数。	Count (カウント)
CmdGet	キャッシュが受信した get コマンドの数。	Count (カウント)
CmdSet	キャッシュが受信した set コマンドの数。	Count (カウント)
CurrConnections	特定の時点でキャッシュに接続された接続回数。ElastiCache 2 ~ 3 つの接続を使用してクラスターを監視します。 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。	Count (カウント)
CurrItems	キャッシュに現在格納されている項目の数。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (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。	Count (カウント)

メトリクス	説明	単位
CmdConfigSet	config set リクエストの累積数。	Count (カウント)
CmdTouch	touch リクエストの累積数。	Count (カウント)
CurrConfig	現在格納されている設定の数。	Count (カウント)
EvictedUnfetched	設定されてからまったくタッチされていない LRU キャッシュから排除された有効な項目の数。	Count (カウント)
ExpiredUnfetched	設定されてからまったくタッチされていない LRU キャッシュから再生された有効期限切れの項目の数。	Count (カウント)
SlabsMoved	移動されたスラブページの合計数。	Count (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。	Count (カウント)
TouchMisses	タッチされたが見つからなかった項目の数。	Count (カウント)

AWS/ElastiCache 名前空間には、計算された以下のキャッシュレベル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。接続は専用であるため、この値は常に 1 以上になります。 ElastiCache	Count (カウント)

メトリクス	説明	単位
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。	Count (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 のオーバーヘッドは、データが使用するメモリに加えてメモリを使用するため、 UnusedMemory このオーバーヘッドを追加データに使用できるメモリ量と見な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。 詳細については、「Memcached item memory usage」を参照してください。	バイト

モニタリングすべきメトリクス

次の CloudWatch メトリクスは、ElastiCache パフォーマンスを把握するのに役立ちます。ほとんどの場合、パフォーマンスの問題が発生する前に修正作業を行うことができるように、これらのメトリクスに CloudWatch アラームを設定することをお勧めします。

モニタリングするメトリクス

- CPUUtilization
- SwapUsage
- Evictions
- CurrConnections

CPUUtilization

パーセン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。

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、より大きい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するか、さらにキャッシュノードを追加してスケールアウトしてください。

SwapUsage

バイト単位でレポートされるホストレベルのメトリクスです。詳細については、「ホストレベルのメトリクス」を参照してください。

`FreeableMemory` CloudWatch メトリクスが 0 に近い (つまり 100 MB 未満) または `SwapUsage` メトリクスが `FreeableMemory` メトリクスより大きい場合、ノードがメモリプレッシャーを受けていることを示します。このような場合には、ConnectionOverhead パラメータ値を大きくすることをお勧めします。

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。

選択したしきい値を超過した場合、大きいノードタイプを使用してクラスターをスケールするか、さらにノードを追加してスケールアウトしてください。

CurrConnections

これは、キャッシュエンジンのメトリクスです。アプリケーションニーズに基づいてこのメトリクスの独自のアラームしきい値を決定することをお勧めします。

CurrConnections の値が大きくなった場合、アプリケーションに問題があることを示している可能性があります。アプリケーション動作を調査してこの問題を解決する必要があります。

CloudWatch クラスターとノードメトリクスのモニタリング

ElastiCache と CloudWatch は、多様なメトリクスを収集できるように統合されています。CloudWatch を使用して、これらのメトリクスをモニタリングできます。

> **Note**
>
> 次の例には、CloudWatch コマンドラインツールが必要です。CloudWatch の詳細と開発者ツールのダウンロードについては、「CloudWatch 製品ページ」を参照してください。

次の手順は、CloudWatch を使用して、過去 1 時間のキャッシュクラスターのストレージ領域統計を収集する方法を示しています。

> **Note**
>
> 以下の例で指定されている `StartTime` 値と `EndTime` 値は、例示を目的としています。実際のキャッシュノードに適した開始時刻値および終了時刻値で置き換える必要があります。

ElastiCache 制限の詳細については、ElastiCache の 「AWS サービス制限」を参照してください。

CloudWatch クラスターとノードメトリクスのモニタリング (コンソール)

キャッシュクラスターの CPU 使用率統計を収集するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. メトリクスを表示するキャッシュノードを選択します。

   > **Note**
   >
   > 20 個を超えるノードを選択すると、コンソールでメトリクスを表示できなくなります。

   a. AWS マネジメントコンソールの [Cache Clusters] ページで、1 つ以上のキャッシュクラスターの名前をクリックします。

      キャッシュクラスターの詳細ページが表示されます。

b. ウィンドウ上部にある [Nodes] タブをクリックします。

c. 詳細ウィンドウの [Nodes] タブで、メトリクスを表示するキャッシュノードを選択します。

使用可能な CloudWatch メトリクスのリストがコンソールウィンドウの下部に表示されます。

d. [CPU Utilization] メトリクスをクリックします。

CloudWatch コンソールが開き、選択されたメトリクスが表示されます。[Statistic] および [Period] ドロップダウンリストボックスや [Time Range] タブを使用すると、表示されるメトリクスを変更できます。

CloudWatch CLI を使用した CloudWatch クラスターとノードメトリクスのモニタリング

キャッシュクラスターの CPU 使用率統計を収集するには

- Linux、macOS、Unix の場合:

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/ElastiCache \
    --metric-name CPUUtilization \
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' \
    --statistics=Average \
    --start-time 2018-07-05T00:00:00 \
    --end-time 2018-07-06T00:00:00 \
    --period=3600
```

Windows の場合:

```
aws cloudwatch get-metric-statistics ^
    --namespace AWS/ElastiCache ^
    --metric-name CPUUtilization ^
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' ^
    --statistics=Average ^
    --start-time 2018-07-05T00:00:00 ^
    --end-time 2018-07-06T00:00:00 ^
    --period=3600
```

CloudWatch API を使用した CloudWatch クラスターとノードメトリックスのモニタリング

キャッシュクラスターの CPU 使用率統計を収集するには

- 以下のパラメータを指定して、CloudWatch API GetMetricStatistics を呼び出します (示されている開始時刻と終了時刻は例です。適切な開始時刻と終了時刻に置き換える必要があります)。
  - Statistics.member.1=Average
  - Namespace=AWS/ElastiCache
  - StartTime=2013-07-05T00:00:00
  - EndTime=2013-07-06T00:00:00
  - Period=60
  - MeasureName=CPUUtilization
  - Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002

  Example

```
http://monitoring.amazonaws.com/
    ?Action=GetMetricStatistics
    &SignatureVersion=4
    &Version=2014-12-01
    &StartTime=2018-07-05T00:00:00
    &EndTime=2018-07-06T23:59:00
    &Period=3600
    &Statistics.member.1=Average
    &Dimensions.member.1="CacheClusterId=mycachecluster"
    &Dimensions.member.2="CacheNodeId=0002"
    &Namespace=&AWS;/ElastiCache
    &MeasureName=CPUUtilization
    &Timestamp=2018-07-07T17%3A48%3A21.746Z
    &AWS;AccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

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、 ElastiCache は特定の Amazon SNSトピックに通知を送信します。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) を使用して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、現在指定されている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)

次の手順は、クラスターに Amazon SNSトピックを追加する方法を示しています。

> **Note**
>
> このプロセスは、Amazon SNSトピックの変更にも使用できます。

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」を選択して、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、以下のパラメータを指定して ModifyCacheClusterアクションを呼び出します。

- `CacheClusterId=my-cluster`
- `TopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications`

Example

```
https://elasticache.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。
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. 通知 のトピック の クラスターの変更 で、通知 を無効にする を選択します。 SNS
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、以下のパラメータmodify-cache-clusterを指定して コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-status inactive
```

Windows の場合:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-status inactive
```

Amazon SNS通知の有効化と無効化 (ElastiCache API)

Amazon SNS通知を無効にするには、以下のパラメータを指定して ModifyCacheClusterアクションを呼び出します。

- CacheClusterId=my-cluster
- NotificationTopicStatus=inactive

この呼び出しにより、以下のような出力が返されます。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
```

```
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicStatus=inactive
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

ElastiCache イベントの表示

ElastiCache は、クラスターのインスタンス、セキュリティグループ、パラメータグループに関連するイベントを記録します。この情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などがあります。ElastiCache コンソール、AWS CLI `describe-events` コマンド、または ElastiCache API アクション `DescribeEvents` を使用して、ログから簡単にイベントを取得できます。

次の手順は、過去 24 時間 (1440 分) のすべての ElastiCache イベントを表示する方法を示しています。

ElastiCache イベントの表示 (コンソール)

次の手順は、ElastiCache コンソールを使用してイベントを表示します。

ElastiCache コンソールを使用してスタックイベントを表示するには

1. AWS Management Console にサインインして、ElastiCache コンソール (https://console.aws.amazon.com/elasticache/) を開きます。
2. 利用可能なすべてのイベントのリストを表示するには、ナビゲーションペインで [Events] を選択します。

   [Events] 画面のリスト内の各行は 1 個のイベントを表し、イベントのソース、イベントの種類 (キャッシュクラスター、キャッシュパラメータグループ、キャッシュセキュリティグループ、キャッシュサブネットグループ)、イベントの GMT 時間、イベントの説明が表示されます。

   [Filter] を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

ElastiCache イベントの表示 (AWS CLI)

AWS CLI を使用して ElastiCache イベントのリストを作成するには、`describe-events` コマンドを使用します。オプションパラメーターを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、最大 40 個のキャッシュクラスターイベントを一覧表示します。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

次のコードでは、過去 24 時間 (1440 分) のすべてのイベントを一覧表示します。

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

describe-events のコマンドによる出力は次のようになります。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
{
    "Events": [
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Finished modifying number of nodes from 1 to 3",
            "Date": "2020-06-09T02:01:21.772Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0002 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.716Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0003 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.706Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Increasing number of requested nodes",
            "Date": "2020-06-09T01:58:34.178Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2c",
            "Date": "2020-06-09T01:51:14.120Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:51:14.095Z"
```

```
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:51:14.094Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:42:55.603Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:42:55.576Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:42:55.574Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:28:40.798Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:28:40.775Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:28:40.773Z"
```

```
        }
    ]
}
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「describe-events」を参照してください。

ElastiCache イベントの表示 (ElastiCache API)

ElastiCache イベントのリストを ElastiCache API を使用して生成するには、DescribeEvents アクションを使用します。オプションパラメーターを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードは、40 個の最新のキャッシュクラスターイベントを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &MaxRecords=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

次のコードは、過去 24 時間 (1440 分) のキャッシュクラスターイベントを一覧します。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

上記のアクションでは、次のような出力が生成されます。

```
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">
    <DescribeEventsResult>
```

```
        <Events>
            <Event>
                <Message>Cache cluster created</Message>
                <SourceType>cache-cluster</SourceType>
                <Date>2015-02-02T18:22:18.202Z</Date>
                <SourceIdentifier>mem01</SourceIdentifier>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「DescribeEvents」を参照してください。

イベント通知と Amazon SNS

ElastiCache 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発行できます。この機能を使用すると、キャッシュクラスターの個々のキャッシュノードエンドポイントに接続されたクライアントコンピュータでサーバーリストを更新できます。

> **Note**
>
> 価格の情報やAmazon SNS ドキュメントへのリンクを含む、Amazon Simple Notification Service (SNS) の詳細については、「Amazon SNS 製品ページ」を参照してください。

通知は、指定した Amazon SNS トピック に発行されます。通知の要件は以下のとおりです:

- ElastiCache 通知用に設定できるトピックは 1 つだけです。
- Amazon SNS トピックを所有する AWS アカウントは、通知が有効になっているキャッシュクラスターを所有するアカウントと同じである必要があります。
- 発行先の Amazon SNS トピックは暗号化できません。

> **Note**
>
> 暗号化された (保存された) Amazon SNS トピックをクラスターにアタッチできます。ただし、 ElastiCache コンソールからのトピックのステータスは非アクティブと表示され、 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の関連付けが効果的に解除されます。

- Amazon SNS トピックは、 ElastiCache クラスターと同じ リージョンに存在する必要があります。

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。

イベント名	メッセージ	説明
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 Virtual Private Cloud (VPC) に新しいキャッシュクラスターに起動する試みが行われました。
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete : 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。 • 1 つ以上の新しい EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループで承認されました。 • 1 つ以上の EC2 セキュリティグループが、キャッシュクラスターに関連付けられたキャッシュセキュリティグループから取り消されました。

イベント名	メッセージ	説明
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache は、キャッシュノードを実行しているホストのパフォーマンスが低下している、または到達できないことを検出し、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 ElastiCache は、 がキャッシュノードを置き換えた後でもキャッシュノードの使用を停止することがあります。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パフォーマンスが低下している、または到達できないことを検出し、キャッシュノードの置き換えを完了しました。 **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 ElastiCache は、 がキャッシュノードを置き換えた後でもキャッシュノードの使用を停止することがあります。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 (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 (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。 実行できるアクションについては、「Memcached のキャッシュノードの交換」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。 実行できるアクションについては、「Memcached のキャッシュノードの交換」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 (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 (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

AWS CloudTrail を使用した Amazon ElastiCache API コール

Amazon ElastiCache は、AWS CloudTrail と統合されています。これは、Amazon ElastiCache のユーザー、ロール、または AWS のサービスで実行されたアクションを記録するためのサービスです。CloudTrail は、Amazon ElastiCache コンソールからの呼び出しと Amazon ElastiCache API オペレーションへのコード呼び出しを含む、Amazon ElastiCache の API コールをイベントとしてキャプチャします。証跡を作成する場合は、Amazon ElastiCache のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、Amazon ElastiCache に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、AWS CloudTrail ユーザーガイドを参照してください。

CloudTrail の Amazon ElastiCache 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。Amazon ElastiCache でアクティビティが発生すると、そのアクティビティは、[Event history] (イベント履歴) にある他の AWS のサービスのイベントとともに、CloudTrail イベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「Viewing Events with CloudTrail Event History」(CloudTrail イベント履歴でのイベントの表示) を参照してください。

Amazon ElastiCache のイベントなど、AWS アカウントのイベントを継続的に記録するには、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 追跡を作成するための概要
- CloudTrail のサポート対象サービスと統合
- Amazon SNS の CloudTrail の通知の設定
- 「複数のリージョンから CloudTrail ログファイルを受け取る」および「複数のアカウントから CloudTrail ログファイルを受け取る」

すべての Amazon ElastiCache アクションは CloudTrail が記録します。これらのアクションは、ElastiCache API リファレンスで説明されています。例えば、CreateCacheCluster、DescribeCacheCluster、ModifyCacheClusterの各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

Amazon ElastiCache ログファイルエントリの理解

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、CreateCacheCluster アクションを示す CloudTrail ログエントリの例です。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:00:35Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"CreateCacheCluster",
    "awsRegion":"us-west-2",
```

```
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "numCacheNodes":2,
        "cacheClusterId":"test-memcached",
        "engine":"memcached",
        "aZMode":"cross-az",
        "cacheNodeType":"cache.m1.small",

    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",

        "cacheClusterStatus":"creating",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "pendingModifiedValues":{
        }
    },
    "requestID":"104f30b3-3548-11e4-b7b8-6d79ffe84edd",
    "eventID":"92762127-7a68-42ce-8787-927d2174cde1"
}
```

次の例は、DescribeCacheCluster アクションを示す CloudTrail ログエントリです。Amazon ElastiCache のすべての Describe 呼び出し (Describe*) について、ResponseElements セクションが削除され、null と表示されます。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:01:00Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"DescribeCacheClusters",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "showCacheNodeInfo":false,
        "maxRecords":100
    },
    "responseElements":null,
    "requestID":"1f0b5031-3548-11e4-9376-c1d979ba565a",
    "eventID":"a58572a8-e81b-4100-8e00-1797ed19d172"
}
```

ModifyCacheCluster アクションを記録する CloudTrail のログエントリの例を以下に示します。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:32:21Z",
    "eventSource":"elasticache.amazonaws.com",
```

```
    "eventName":"ModifyCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "applyImmediately":true,
        "numCacheNodes":3,
        "cacheClusterId":"test-memcached"
    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "cacheClusterCreateTime":"Dec 1, 2014 10:16:06 PM",
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",
        "cacheClusterStatus":"modifying",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "configurationEndpoint":{
            "address":"test-memcached.example.cfg.use1prod.cache.amazonaws.com",
            "port":11211
        },
        "pendingModifiedValues":{
            "numCacheNodes":3
        }
    },
    "requestID":"807f4bc3-354c-11e4-9376-c1d979ba565a",
    "eventID":"e9163565-376f-4223-96e9-9f50528da645"
```

```
}
```

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	デフォルト値
リージョンごとのサーバーレスキャッシュ	40
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	60
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

リファレンス

このセクションのトピックでは、Amazon ElastiCache API および AWS CLI の ElastiCache セクションの使用について説明します。また、このセクションには一般的なエラーメッセージとサービス通知も含まれます。

- ElastiCache API の使用
- ElastiCache API リファレンス
- AWS CLI リファレンスの ElastiCache セクション
- Amazon ElastiCache エラーメッセージ
- 通知

ElastiCache API の使用

このセクションでは、ElastiCache のオペレーションを使用および実装する方法を、メソッドに重点を置いて説明します。これらのオペレーションの詳細については、「Amazon ElastiCache API リファレンス」を参照してください。

トピック

- クエリ API を使用する
- 利用可能なライブラリ
- アプリケーションのトラブルシューティング

クエリ API を使用する

クエリパラメータ

HTTP クエリベースのリクエストとは、HTTP 動詞 (GET または POST) とクエリパラメータ `Action` で記述する HTTP リクエストです。

各クエリリクエストに、アクションの認証と選択を処理するための一般的なパラメータがいくつか含まれている必要があります。

オペレーションの中にはパラメータのリストを取るものがあります。これらのリストは、`param.`*n* 表記を使用して指定されます。*n* 値は、1 から始まる整数です。

クエリリクエストの認証

HTTPS 経由でのみリクエストを送信できます。また、各クエリリクエストには署名を含める必要があります。このセクションでは、署名を作成する方法について説明します。次に説明する方法は、署名バージョン 4 と呼ばれます。

AWS へのリクエストを認証するために使用される基本的な手順を次に示します。この手順では、AWS に登録されており、アクセスキー ID とシークレットアクセスキーを持っていることを前提としています。

クエリ認証プロセス

1. 送信者は、AWS へのリクエストを構築します。
2. このトピックの次のセクションに示すように、送信者は、SHA-1 ハッシュ関数を使用してリクエストの署名 (Hash-based Message Authentication Code (HMAC) のキー付きハッシュ) を生成します。
3. リクエストの送信者は、リクエストデータ、署名、およびアクセスキー ID (使用するシークレットアクセスキーのキー識別子) を AWS に送信します。
4. AWS ではアクセスキー ID を使用して、シークレットアクセスキーを調べます。
5. AWS では、リクエストの署名を生成する際に使用したものと同じアルゴリズムを使い、リクエストデータとシークレットアクセスキーから署名を生成します。
6. 署名が一致すると、リクエストは認証されたものと見なされます。もし署名が一致しなかった場合、リクエストの処理は拒否され、AWS はエラーレスポンスを返します。

> **Note**
>
> リクエストに `Timestamp` パラメータが含まれている場合、リクエストに対して生成された署名はパラメータの値の 15 分後に期限が切れます。
> リクエストに `Expires` パラメータが含まれている場合、署名は `Expires` パラメータで指定された時刻に期限が切れます。

リクエストの署名を計算するには

1. 本手順で後に必要となる、正規化されたクエリ文字列を作成します。

a. 自然なバイト順のパラメータ名で、UTF-8 のクエリ文字列コンポーネントを並び替えます。パラメータは、GET URI または POST ボディから取得される場合があります。(Content-Type が application/x-www-form-urlencoded の場合)

b. URL は、以下の規則に応じてパラメータ名と値をエンコードします。

   i. RFC 3986 が定義する非予約文字を、URL がエンコードすることはありません。非予約文字とは、A〜Z、a〜z、0〜9、ハイフン (-)、アンダーバー (_)、ピリオド (.)、およびチルド (~) です。

   ii. 他のすべての文字についても、%XY (X および Y には HEX 文字の 0-9 および大文字の A-F が入る) によるパーセントエンコードが必要です。

   iii. パーセントは、拡張 UTF-8 文字を %XY%ZA.... 形式でエンコードします。

   iv. パーセントは、スペース文字を %20 (通常エンコードスキーマが行なうような + ではありません) としてエンコードします。

c. パラメータの値が空値の場合でも、エンコードされるパラメータ名とエンコードされる値の間に等号 (=) (ASCII コード 61) を入れます。

d. それぞれのパラメータ名と値のペアをアンド (&) (ASCII コード 38) で分割します。

2. 文字列を作成し、以下の擬似文法に従って ("\n" は ASCII 新規行を意味します) 署名を作成します。

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

   HTTPRequestURI 要素は URI の HTTP 絶対パス要素ですが、クエリ文字列は含みません。HTTPRequestURI が空値の場合は、スラッシュ (/) を使用してください。

3. 作成したばかりの文字列を使い、シークレットアクセスキーをキーとして、また SHA256 または SHA1 をハッシュアルゴリズムとして、RFC 2104 に準拠した HMAC を計算します。

   詳細については、https://www.ietf.org/rfc/rfc2104.txt を参照してください。

4. 結果の値を base64 に変換します。

5. その値は、`Signature` パラメータの値としてリクエストに含めます。

サンプルのリクエストを次に示します (見やすくするために改行が追加されています)。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
```

前のクエリ文字列では、次の文字列に対する HMAC 署名が生成されます。

```
GET\n
    elasticache.amazonaws.com\n
    Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-west-2%2Felasticache
%2Faws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:elasticache.us-west-2.amazonaws.com
        user-agent:CacheServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

結果の署名付きリクエストは次のようになります。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-west-2/elasticache/aws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

プロセスへの署名とリクエスト署名の生成の詳細については、トピック「署名バージョン 4 の署名プロセス」とそのサブトピックを参照してください。

利用可能なライブラリ

AWS では、クエリ API の代わりに言語固有の API を使用してアプリケーションを構築するソフトウェア開発者向け Software Development Kit (SDK) を提供します。こうした SDK には、リクエスト認証、リクエストの再実行、エラー処理など、 (API には含まれない) 基本的な機能が用意されていて、簡単に開始できるようになっています。次のプログラミング言語の SDK と追加のリソースがあります。

- Java
- Windows および .NET
- PHP
- Python
- Ruby

他の言語については、「サンプルコードとライブラリ」を参照してください。

アプリケーションのトラブルシューティング

ElastiCache では、ElastiCache API とのやり取りで発生する問題をトラブルシューティングする際に役立つ、具体的でわかりやすいエラーを提供します。

エラーの取得

通常、アプリケーションでは、結果を処理する前にリクエストでエラーが生成されたかどうかを必ず確認します。エラーが発生したかどうかを確認する最も簡単な方法は、ElastiCache API からのレスポンスで `Error` ノードを検索することです。

XPath 構文を使用すると、簡単な方法で `Error` ノードがあるかどうかを検索し、エラーコードとメッセージを取得することができます。次のコードでは、Perl および XML::XPath モジュールによって、リクエスト時のエラーの発生を判定しています。エラーが発生した場合、レスポンス内の最初のエラーコードとメッセージが表示されます。

```
use XML::XPath;
```

```
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$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

ElastiCache API の問題を診断して解決するには、次の手順を実行することをお勧めします。

- ElastiCache が正しく実行されていることを確認します。

  これを行うには、ブラウザウィンドウを開いて、ElastiCache サービス (https://elasticache.amazonaws.com など) に対してクエリリクエストを送信します。MissingAuthenticationTokenException または内部サーバーエラー 500 は、サービスが利用可能であり、リクエストに応答していることを示します。

- リクエストの構文を確認します。

  「ElastiCache API リファレンス」には、各 ElastiCache オペレーションについてのリファレンスページがあります。パラメータを正しく使用していることをもう一度確認してください。間違っている可能性がある部分を判断するヒントとして、同様のオペレーションを実行しているサンプルのリクエストやユーザーシナリオを調べてください。

- フォーラムを確認します。

  ElastiCache にはディスカッションフォーラムがあります。このフォーラムでは、これまで他のユーザーが経験してきた問題に対する解決策を探すことができます。フォーラムを見るには、以下をご覧ください。

  https://forums.aws.amazon.com/

ElastiCache コマンドラインインターフェイスの設定

このセクションでは、コマンドラインツールを実行するための前提条件、コマンドラインツールを入手できる場所、ツールの設定方法と環境について説明します。また、このセクションにはツール使用の一般的な例も含まれています。

AWS CLI for ElastiCache を使用する場合にのみ、このトピックの手順に従います。

> **Important**
>
> Amazon ElastiCache コマンドラインインターフェイス (CLI) は、API バージョン 2014-09-30 以降の ElastiCache の機能強化をサポートしていません。コマンドラインから新しい ElastiCache 機能を使用するには、AWS コマンドラインインターフェイスを使用します。

トピック

前提条件

この文書では、Linux/UNIX または Windows 環境での作業が可能であることを想定しています。Amazon ElastiCache コマンドラインツールは、UNIX ベースの環境である Mac OS X でも動作しますが、このガイドには Mac OS X に関する固有の手順は含まれていません。

慣例として、すべてのコマンドラインテキストには、一般的な **PROMPT>** コマンドラインプロンプトが前に付けられています。マシンの実際のコマンドラインプロンプトは、異なっている可能性があります。また、Linux/UNIX 固有のコマンドを示すには、**$** 、Windows 固有のコマンドを示すには、**C:\>** を使用します。コマンドからの出力例は、その直後にプレフィックスなしで表示されています。

Java ランタイム環境

このガイドで使用されているコマンドラインツールを実行するには、Java バージョン 5 以降が必要です。JRE または JDK のどちらでもかまいません。Linux/UNIX と Windows を含め各種プラットフォーム用の JRE は、Java SE ダウンロードから表示してダウンロードすることができます。

Java Home 変数の設定

コマンドラインツールは、Java ランタイムの場所を特定する環境変数 (`JAVA_HOME`) に応じて異なります。この環境変数は、`bin` 実行可能ファイル (Linux および UNIX の場合) または `java` 実行可

能ファイル (Windows の場合) が存在する java.exe という名前のサブディレクトリが含まれているディレクトリの絶対パスに設定する必要があります。

JAVA_HOME 変数の設定方法

1. Java ホーム変数を設定します。

   - Linux と UNIX では、以下のコマンドを入力します。

   ```
   $ export JAVA_HOME=<PATH>
   ```

   - Windows は、以下のコマンドを入力します。

   ```
   C:\> set JAVA_HOME=<PATH>
   ```

2. **$JAVA_HOME/bin/java -version** を実行して出力をチェックすることにより、パス設定を確認します。

   - Linux/UNIX では、以下のような出力結果が表示されるはずです。

   ```
   $ $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 開発者ツールウェブサイト」で ZIP ファイルとして入手できます。このツールは Java で作成されており、Windows 2000/XP/Vista/Windows 7 と Linux/UNIX、および Mac OSX 用のシェルスクリプトが含まれています。ZIP ファイルは自己完結型であ

り、インストール作業は不要です。ZIP ファイルをダウンロードして圧縮を解除し、ローカルマシン上の任意のディレクトリに保存してください。

ツールを設定する

コマンドラインツールは、サポートライブラリの場所の特定を環境変数 (AWS_ELASTICACHE_HOME) に依存しています。ツールを使うには、この環境変数を設定する必要があります。コマンドラインツールを解凍したディレクトリのパスに設定します。このディレクトリの名前は、ElastiCacheCli-A.B.nnnn (A、B、および n はバージョン/リリース番号) で、bin および lib という名前のサブディレクトリが含まれます。

AWS_ELASTICACHE_HOME 環境変数を設定するには

- コマンドラインウィンドウを開き、次のいずれかのコマンドを入力して、AWS_ELASTICACHE_HOME 環境変数を設定します。
  - Linux と UNIX では、以下のコマンドを入力します。

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

  - Windows は、以下のコマンドを入力します。

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

ツールを使いやすくするため、ツールの BIN ディレクトリをシステム PATH に追加することをお勧めします。このガイドの残りの部分は、BIN ディレクトリがシステムパスに含まれていることを前提としています。

ツールの BIN ディレクトリをシステムパスに追加するには

- システム PATH にツールの BIN ディレクトリを追加するには、次のコマンドを入力します。
  - Linux と UNIX では、以下のコマンドを入力します。

```
$ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
```

  - Windows は、以下のコマンドを入力します。

```
C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
```

> **Note**
>
> Windows の環境変数は、コマンドウィンドウを閉じたときにリセットされます。永続的に設定することもできます。詳細については、使用している Windows のバージョンのドキュメントを参照してください。

> **Note**
>
> スペースを含むパスは、二重引用符で囲む必要があります。たとえば、次のとおりです。「C:\Program Files\Java」

ツールでの認証情報の指定

コマンドラインツールを使用するには、AWS アカウントで提供される AWS アクセスキーとシークレットアクセスキーが必要です。コマンドラインを使用するか、ローカルシステムにある認証情報ファイルから取得できます。

環境には、情報の編集に必要なテンプレートファイル ${AWS_ELASTICACHE_HOME}/credential-file-path.template が含まれています。テンプレートファイルの内容は次のとおりです。

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

> **Important**
>
> UNIX では、アクセス許可を認証情報ファイルの所有者に制限します。
>
> ```
> $ chmod 600 <the file created above>
> ```

認証情報ファイルを設定したら、ElastiCache ツールで情報が検索されるように AWS_CREDENTIAL_FILE 環境変数を作成する必要があります。

AWS_CREDENTIAL_FILE 環境変数の設定方法

1. 環境変数を設定します:

   - Linux/UNIX では、次のコマンドを使用して変数を更新します。

     ```
     $ export &AWS;_CREDENTIAL_FILE=<the file created above>
     ```

   - Windows では、次のコマンドを使用して変数を設定します。

     ```
     C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
     ```

2. 設定が適切に機能することをチェックし、次のコマンドを実行します。

   ```
   elasticache --help
   ```

   すべての ElastiCache コマンドの使用方法ページが表示されます。

環境変数

環境変数はスクリプトやデフォルト値の設定またはそれらを一時的に上書きする場合に便利です。

AWS_CREDENTIAL_FILE 環境変数に加えて、ElastiCache コマンドラインインターフェイスに付属するほとんどの API ツールは次の変数をサポートしています。

- EC2_REGION — 使用する AWS リージョンです。
- AWS_ELASTICACHE_URL — サービス呼び出しに使用する URL です。EC2_REGION が指定されている場合または -- リージョンパラメータが渡されている場合は、別のリージョンのエンドポイントを指定する必要はありません。

以下の例は API ツールで使用するリージョンを設定する環境変数 EC2_REGION の設定方法を示します。

Linux、OS X、Unix

```
$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

Amazon ElastiCache エラーメッセージ

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 のサービス ElastiCache、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
- 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 の制限」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 の制限」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: 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 (Memcached)」を参照してください。

通知

このトピックでは、関心のある 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。

通知では、新機能や改善された ElastiCache 機能については発表されません。

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 エンジンに固有です。

ElastiCache (Memcached) 固有の通知

- アラート: Memcached の LRU クローラによりセグメント障害が生じます

アラート: Memcached の LRU クローラによりセグメント障害が生じます

アラート日付: 2017 年 2 月 28 日
特定の状況では、クラスターは Memcached LRU クローラで、セグメント障害による不安定な状態を表示する場合があります。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。問題は LRU クローラがデフォルトで有効化されたときに、Memcached 1.4.33 で明確になりました。
この問題が発生した場合、修正されるまで LRU クローラ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:

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 (Memcached) 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 (Memcached) 更新

変更	説明	日付
ElastiCache (Memcached) 1.6.22 のサポート	ElastiCache (Memcached) で Memcached 1.6.22 がサポートされるようになりました。また、バージョン 1.6.18 の累積バグ修正が含まれています。詳細については、「Memcached バージョン 1.6.22」を参照してください。	2024 年 1 月 10 日
ElastiCache (Memcached) でサーバーレスキャッシュの作成がサポートされ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可	2023 年 11 月 27 日

	能にする新しいアクセス許可AmazonElastiCacheFullAccess 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) 1.6.17 のサポート	ElastiCache (Memcached) で Memcached 1.6.17 がサポートされるようになりました。これには、Memcached バージョン 1.6.12 からの累積的なバグ修正が含まれています。詳細については、「Memcached バージョン 1.6.17」を参照してください。	2023 年 1 月 18 日

ElastiCache (Memcached) がをサポートするようになりました IPV6	ElastiCache はインターネットプロトコルバージョン 4 および 6 (IPv4 および IPv6) をサポートしているため、IPv4接続のみ、IPv6接続のみ、または IPv4および IPv6 接続 (デュアルスタック) の両方を受け入れるようにクラスターを設定できます。IPv6は、Nitro System 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) が転送時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 「転送時の暗号化 (TLS) 」を参照してください。	2022 年 5 月 26 日

ElastiCache が をサポートするようになりました PrivateLink	AWS PrivateLink を使用すると、インターネットゲートウェイ、NATデバイス、VPN接続、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 用の Amazon ElastiCache API とインターフェイスVPCエンドポイント (AWS PrivateLink) OSS」または「Memcached 用の Amazon ElastiCache API とインターフェイスVPCエンドポイント (AWS PrivateLink) 」を参照してください。	2022 年 1 月 24 日
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。 では、条件キーのサポート ElastiCache も導入されています。IAM ポリシーを有効にする方法を決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

Memcached 1.6.6 がサポートされました。	ElastiCache (Memcached) で Memcached 1.6.6 がサポートされるようになりました。また、バージョン Memcached 1.5.16 の累積バグ修正が含まれています。詳細については、「Memcached バージョン 1.6.6」を参照してください。	2020 年 11 月 12 日
ElastiCache が Outposts AWS で利用可能になりました	AWS Outposts は、ネイティブ AWS サービス、インフラストラクチャ、運用モデルを事実上あらゆるデータセンター、コロケーションスペース、またはオンプレミス施設に導入します。Outposts ElastiCache にデプロイすると、クラウドと同じようにオンプレミスでキャッシュをセットアップ、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 でローカルゾーンがサポートされ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、任意の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 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 が ElastiCache クラスターの自動更新をサポート	Amazon は、サービス更新の「推奨適用日」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。 は、メンテナンスウィンドウ ElastiCache 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

Amazon ElastiCache が T3-Standardキャッシュノードをサポート	Amazon で次世代の汎用バースト T3-Standardキャッシュノードを起動できるようになりました ElastiCache。Amazon EC2の T3-Standardインスタンスは、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使用量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
ElastiCache (Memcached) では、ユーザーが自分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、特にピーク時のビジネスフロー中のサービスの中断を最小限に抑えることができ、セキュリティ更新によってコンプライアンスを確保できます。詳細については、「Amazon でのセルフサービスの更新 ElastiCache」を参照してください。	2019 年 10 月 9 日

ElastiCache (Memcached) 1.5.16 のサポート	ElastiCache (Memcached) で Memcached 1.5.16 がサポートされるようになりました。また、バージョン Memcached 1.5.14 および Memcached 1.5.15 の累積バグ修正が含まれています。詳細については、「Memcached バージョン 1.5.16」を参照してください。	2019 年 9 月 6 日
ElastiCache スタンダードリザーブドインスタンスの提供: 一部前払い、全額前払い、全額前払いなし。	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 (Memcached) 1.5.10 のサポート	ElastiCache (Memcached) で Memcached 1.5.10 がサポートされるようになりました。no_modern および inline_ascii_resp パラメータのサポートが含まれています。詳細については、「Memcached バージョン 1.5.10」を参照してください。	2018 年 11 月 14 日

ユーザーガイドの再編成	1 つのElastiCache ユーザーガイドが再構成され、Redis OSS (ElastiCache (Redis OSS) 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分離され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

次の表は、2018 年 3 月以前の ElastiCache (Memcached) 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 は、アジアパシフィック (大阪ローカル) リージョンのサポートを追加しました。現在、アジアパシフィック (大阪) リージョンでは、1 つのアベイラビリティ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 • サポートされているキャッシュノードの種類	2018 年 2 月 12 日
欧州 (パリ) のサポート	ElastiCache は、欧州 (パリ) リージョンのサポートを追加しました。詳細については、次を参照してください。 •	2017 年 12 月 18 日

変更	説明	変更日
	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	
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、サービスにリンクされたロール () 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 ElastiCache でのサービスにリンクされたロールの使用 • アクセス許可を設定する (新規 ElastiCache ユーザーのみ)	2017 年 12 月 7 日
R4 ノードタイプのサポート	ElastiCache の今回のリリースでは、 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	2017 年 11 月 20 日

変更	説明	変更日
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Amazon VPC 内の ElastiCache キャッシュにアクセスするためのアクセスパターン」のElastiCache 「」を参照してください。	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が組み込まれている Memcached バージョン 1.4.34 のサポートを追加します。 詳細については、 の Memcached 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
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 で、欧州 (ロンドン) リージョンのサポートが追加されました。現在、T2 および M4 のノードタイプのみサポートされています。詳細については、次を参照してください。 • 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

変更	説明	変更日
カナダ (モントリオール) リージョンのサポート	ElastiCache では、カナダ (モントリオール) リージョンのサポートが追加されました。現在、このAWS リージョンではノードタイプ M4 と T2 のみがサポートされています。詳細については、次を参照してください。 • 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 では、南米 (サンパウロ) リージョンの R3 および M4 ノードタイプと、中国 (北京) リージョンの M4 ノードタイプのサポートが追加されました。詳細については、次を参照してください。 • サポートされるリージョン • サポートされているキャッシュノードの種類	2016 年 11 月 1 日
米国東部 2 (オハイオ) リージョンのサポート	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
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	2016 年 8 月 3 日

変更	説明	変更日
ムンバイリージョンのサポート	ElastiCache では、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	2016 年 6 月 27 日
R3 ノードタイプのサポート	ElastiCache は、中国 (北京) リージョンと南米 (サンパウロ) リージョンで R3 ノードタイプのサポートを追加します。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 へのアクセス	Amazon で にアクセスするように Lambda 関数を設定するチュートリアルを追加 ElastiCache しましたVPC。詳細については、「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
アジアパシフィック (ソウル) リージョンのサポート	ElastiCache は、t2、m3、および 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を追加します。	2016 年 1 月 6 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されていない (LRU) キャッシュ管理、ハッシュアルゴリズムとしての 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

変更	説明	変更日
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、バージョン 5.6 の Memcached Auto Discovery PHP 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、「PHP 向けの 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」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
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: に関する 3 つのメッセージを追加しますNodeReplacementRescheduledElastiCacheNodeReplacementCanceled。 ノードの置き換え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
コスト配分タグのサポート	ElastiCache では、コスト配分タグのサポートが追加されました。詳細については、「コスト配分タグによるコストのモニタリング」を参照してください。	2015 年 2 月 9 日
AWS GovCloud （米国西部) リージョンのサポート	ElastiCache 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 では、欧州 (フランクフルト) (eu-central-1) リージョンのサポートが追加されました。	2015 年 1 月 19 日

変更	説明	変更日
AWS CloudTrail サポートされているAPI呼び出しのログ記録	ElastiCache では、 を使用してすべての ElastiCache API呼び出し AWS CloudTrail を ログに記録するためのサポートが追加されました。詳細については、「AWS CloudTrail を使用した Amazon ElastiCache API コール」を参照してください。	2014 年 9 月 15 日
サポートされる新しいインスタンスサイズ	ElastiCache 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
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
デフォルトの Amazon Virtual Private Cloud のサポート (VPC)	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。新規のお客様の場合、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 VPC と ElastiCache のセキュリティ」を参照してください。	2013 年 1 月 8 日

変更	説明	変更日
PHP キャッシュノード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 へのキャッシュノードの自動検出サポートを提供しますPHP。	2013 年 1 月 2 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 VPC と ElastiCache のセキュリティ」を参照してください。	2012 年 12 月 20 日
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。これは、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決定し、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。 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

変更	説明	変更日
リザーブドキャッシュノード	このリリースは、リザーブドキャッシュノードのサポートを追加します。	2012 年 4 月 5 日
新規ガイド	これは Amazon ElastiCache ユーザーガイド の最初のリリースです。	2011 年 8 月 22 日

AWS 用語集

最新の AWS 用語については、「 AWS の用語集 リファレンス」のAWS 「 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。