での Microsoft ワークロードのコストの最適化 AWS

AWS 規範ガイダンス

AWS 規範ガイダンス: での Microsoft ワークロードのコストの最適化 AWS

Table of Contents

での Microsoft ワークロードのコストを最適化する AWS

Bill Pfeiffer、Chase Lindeman、Kevin Sookhan、Amazon Web Services (AWS)

2025 年 10 月 (ドキュメント履歴)

概要:

このガイドでは、Microsoft ワークロードのコストを最適化するための推奨事項、ベストプラクティス、戦略について説明します AWS。このガイドには、ビジネス目標を達成する費用対効果の高い高性能ワークロードの構築と自動化に役立つ基本的な AWS の知識、コスト最適化手法、リファレンスアーキテクチャも含まれています。総称して、このガイダンスは Microsoft on AWS Cost Optimization (MACO) と呼ばれます。MACO ガイダンスは業界の専門家によって開発され、現実世界のシナリオに基づいています。

このガイドでは、以下の Microsoft ワークロードについて説明します。

- Amazon Elastic Compute Cloud (Amazon EC2) の Windows
- SQL Server
- コンテナ
- Storage
- Active Directory
- .NET

オーディエンス

このガイドは、アーキテクト、エンジニア、管理者、ディレクター、CTOs、技術意思決定者、AWS パートナーを対象としています。は役に立ちますが、 AWS 請求、Microsoft テクノロジー、AWS システム管理に関する経験があり、基本的な知識を持っている必要はありません。

このガイドの使い方

このガイドを使用して、クラウドへの MACO ジャーニーを計画および実装できます。このガイドを最初から最後まで読んで、Microsoft ワークロードのコストを最適化するためのオプションとアプローチを包括的に理解することをお勧めします AWS。組織のニーズに基づいて、次のワークロードセクションを確認できます。

- Amazon EC2 の Windows
- SQL Server
- コンテナ
- [Storage (ストレージ)]
- Active Directory
- .NET

> **Important**
>
> このガイドで提供されるコードのサンプルは、デモンストレーションのみを目的としています。ベストプラクティスは、本番環境で使用する前に、開発環境ですべてのコードをテストすることです。コードを実装する前に、コードを小さなバッチでテストし、AWS Cost Explorer を使用してコードによって生じるコストの変化を確認することをお勧めします。これは、後で問題になる可能性のあるエッジケースやその他の問題のトラブルシューティングに役立ちます。

> **Important**
>
> このガイドの料金例は、公開時の価格に基づいています。価格は変更されることがあります。さらに、コストは、 AWS リージョン、 AWS のサービス クォータ、およびクラウド環境に関連するその他の要因によって異なる場合があります。

目標とするビジネス成果

このガイドは、次に示す、お客様やお客様組織のビジネス成果実現に有用です。

- AWS Optimization and Licensing Assessment (AWS OLA) を使用して、リソース使用率、サードパーティーのライセンス、およびアプリケーションの依存関係に基づいて、現在のオンプレミス環境とクラウド環境を評価および最適化する方法を学びます。
- Microsoft ワークロード用の AWS モダナイゼーション計算ツールを使用して、コスト最適化のビジネスケースを作成します。
- Amazon Elastic Compute Cloud (Amazon EC2)、SQL Server、コンテナ、ストレージ、Active Directory、および .NET 上の Windows のワークロードを含む、特定の Microsoft ワークロードのコストを最適化します。

コスト最適化ジャーニー

クラウド移行ジャーニーの範囲、タイミング、具体的な道筋は、ビジネス目標、技術要件、その他の要因によって異なります。このセクションでは、AWS を使用したクラウド財務管理に焦点を当て、MACO の推奨事項とベストプラクティスに準拠したクラウド移行ジャーニーの例を示します。この例を使用して、Microsoft ワークロードのクラウド移行ジャーニーを設計する方法を理解できます。

以下の高レベルタスクは、組織が MACO の推奨事項とベストプラクティスを実装するために実行できるアプローチを示しています。

- タグ付け戦略を確立し、ユーザー定義のコスト配分タグを有効にします。詳細については、AWS ホワイトペーパー「Best Practices for Tagging AWS Resources」を参照してください。
- アプリケーション、チーム、または部門に基づいて予算を定義します。詳細については、「AWS 請求とコスト管理ユーザーガイド」の「AWS Budgets を使用したコストの管理」を参照してください。
- AWS Optimization and Licensing Assessment (AWS OLA) を実行して、コスト削減を加速します。詳細については、AWS ドキュメントの「AWS Optimization and Licensing Assessment」を参照してください。
- Amazon Elastic Compute Cloud Dedicated Hosts を使用した、Windows および SQL Server ワークロード用の Bring Your Own License (BYOL)。詳細については、このガイドの「Windows および SQL Server ワークロードのライセンスを持ち込む」セクションを参照してください。
- AWS で SQL Server ライセンスを最適化します。詳細については、このガイドの「SQL Server ライセンスを理解する」セクションを参照してください。
- Windows ワークロードに適したインスタンスタイプを選択します。詳細については、このガイドの「Windows ワークロードに適したインスタンスタイプを選択する」セクションを参照してください。
- SQL ワークロードに適したインスタンスタイプを選択します。詳細については、このガイドの「SQL Server ワークロードに適した EC2 インスタンスを選択する」セクションを参照してください。
- Amazon Elastic Block Store (Amazon EBS) を gp2 から gp3 に移行します。詳細については、このガイドの「Amazon EBS ボリュームを gp2 から gp3 に移行する」セクションを参照してください。
- AWS で EC2 Instance Scheduler を使用してワークロードを制御します。詳細については、このガイドの「停止スケジュールと開始スケジュールを自動化する」セクションを参照してください。

- SQL Server Developer Edition を使用して、非本番ワークロードの SQL Server コストを削除します。詳細については、このガイドの「SQL Server Developer Edition を評価する」セクションを参照してください。
- 開発およびテストワークロードには、Amazon FSx for Windows File Server 用の単一のアベイラビリティーゾーンを使用します。詳細については、このガイドの「単一のアベイラビリティーゾーンを使用する」セクションを参照してください。
- AWS Compute Optimizer を使用して、Windows ワークロードを適切なサイズに設定します。詳細については、このガイドの「Windows ワークロードを適切なサイズに設定する」セクションを参照してください。
- Savings Plans を使用して、Amazon EC2 上の Windows の支出を最適化します。詳細については、このガイドの「Amazon EC2 での Windows の支出を最適化する」セクションを参照してください。
- FSx for Windows File Server でデータ重複排除を有効にします。詳細については、このガイドの「Amazon FSx でデータ重複排除を有効にする」セクションを参照してください。
- FSx for Windows File Server のファイルシステムにデータシャーディングを使用します。詳細については、このガイドの「FSx for Windows File Server のデータシャーディングを理解する」セクションを参照してください。
- SQL Server のバックアップ戦略を最適化します。詳細については、このガイドの「SQL Server バックアップ戦略を最適化する」セクションを参照してください。
- 静的 .NET Framework アプリが動的スケーリングをサポートするようにします。詳細については、このガイドの「静的 .NET Framework アプリケーションの動的スケーリングをサポートする」を参照してください。
- サーバーレス .NET マイクロサービスを使用します。詳細については、このガイドの「サーバーレス .NET を検討する」セクションを参照してください。
- Windows アプリをコンテナに移動します。詳細については、このガイドの「.NET アプリケーションをコンテナ化する」セクションを参照してください。
- AWS Compute Optimizer を使用して、AWS Fargate for Amazon Elastic Container Service (Amazon ECS) で実行されている Windows コンテナのサイズを適切に設定します。詳細については、このガイドの「Compute Optimizer を有効にする」セクションを参照してください。
- 最新の .NET にリファクタリングし、Linux に移行します。詳細については、このガイドの「最新の .NET にリファクタリングして Linux に移行する」セクションを参照してください。
- Graviton インスタンスとコンテナを活用します。詳細については、このガイドの「Graviton インスタンスとコンテナを使用する」セクションを参照してください。

- SQL Server データベースをモダナイズします。詳細については、このガイドの「SQL Server データベースをモダナイズする」セクションを参照してください。
- Active Directory インフラストラクチャを設計します。詳細については、このガイドの「Active Directory」セクションを参照してください。

AWS を使用したクラウド財務管理に焦点を当てたカスタマージャーニーの詳細については、AWS ホワイトペーパーの「Cloud Financial Management capability」を参照してください。

コストを最適化するための上位の推奨事項

概要

コスト最適化は AWS Well-Architected フレームワークの柱の 1 つであり、クラウド移行計画において重要な役割を果たします。このガイド全体を通してコスト最適化の推奨事項を確認できますが、このセクションでは最も影響の大きい推奨事項について説明します。これらの推奨事項は迅速に実装でき、組織に大きな影響を与えることになります。これらの推奨事項は、コスト最適化作業全体の基盤を構築するのに役立ちます。

上位の推奨事項

次の表に、最も影響の大きいコスト最適化に関する上位の推奨事項を示します。「実装の難しさ」列では、実装が最も簡単なもの (1) から実装が最も難しいもの (5) までのスケールに基づいて、各最適化を評価しています。「推定削減額」列には、推奨される最適化ごとに組織が削減できる額のパーセンテージベースの見積もりが表示されます。

最適化	実装の難しさ	推定削減額
Windows ワークロードを適切なサイズに設定する	3	25%
Windows および SQL Server ワークロードのライセンスを持ち込む	3	30%
SQL Server Developer Edition を評価する	2	20%
SQL Server ライセンスを理解する	2	最大 50%
停止スケジュールと開始スケジュールを自動化する	3	最大 40%

最適化	実装の難しさ	推定削減額
Windows ワークロードに適したインスタンスタイプを選択する	1	10～30%
最新の .NET にリファクタリングして Linux に移行する	5	10～20%
Amazon EC2 での Windows の支出を最適化する	3	最大 20～40%
Amazon EBS ボリュームを gp2 から gp3 に移行する	4	最大 20%

Important

上記の表の推定削減額は、アカウント内の全体的な AWS の支出ではなく、個々の技術ドメインに適用されます。例えば、Instance Scheduler をさまざまな環境タイプとサイズで実装することで、可能性のある削減額を変えることができます。見積もりは、特に Amazon EC2 インスタンスのコストに適用され、他の AWS のサービス の全体的なコスト削減を意味するものではありません。これらの見積もりはゲージとして提供されており、保証するものではありません。

MACO エキスパートは、コスト最適化についてより詳細に説明できます。ユースケースを深く掘り下げるためのミーティングを設定するには、アカウントチームに連絡するか、optimize-microsoft@amazon.com まで E メールでご連絡ください。

AWS 最適化とライセンス評価

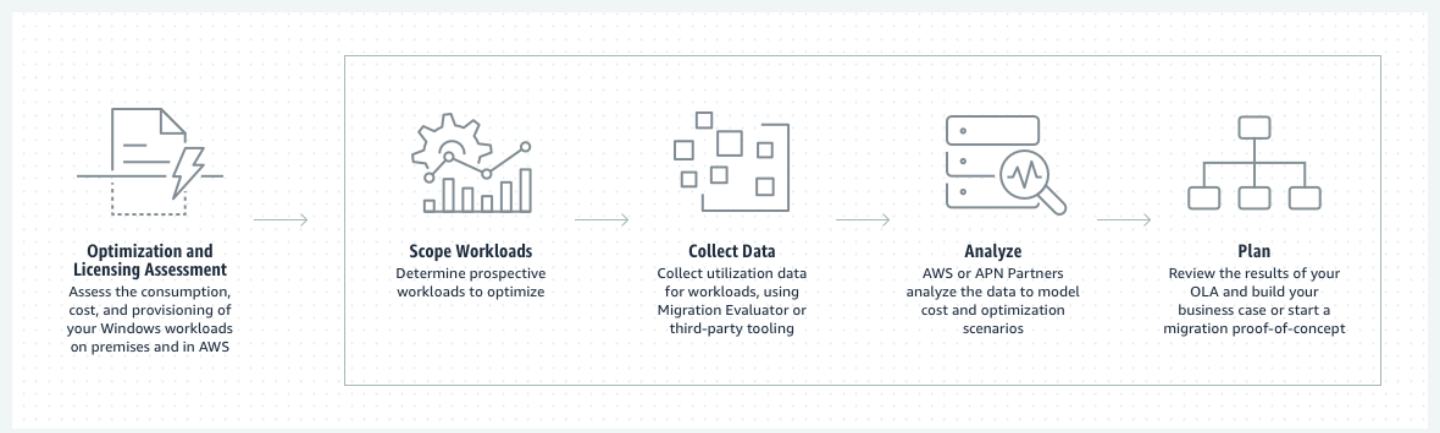
概要

AWS Optimization and Licensing Assessment (AWS OLA) を使用すると、リソース使用率、サードパーティのライセンス、およびアプリケーションの依存関係に基づいて、現在のオンプレミス環境と既存のクラウド環境を評価および最適化できます。AWS OLA を使用すると、組織は、AWS に移行したり、AWS 上の既存の Microsoft ワークロードを評価したりする際のコスト削減を可能にする移行およびライセンス戦略を構築できます。AWS OLA は、以下の達成にも役立ちます。

- 既存のデプロイ、アプリケーションのパフォーマンス、および契約について理解する。
- リソースのサイズを適正化する。
- AWS クラウド へのロードマップを作成する。
- 既存の投資を使用し、使用した分だけ支払うことで、コストを削減する。

AWS OLA をコスト最適化ジャーニーの最初のステップにすることをお勧めします。AWS Partner Network を使用して AWS OLA を完了できます。これらは、評価データを収集し、ライセンスとインスタンスのコストを最適化するための推奨事項を提供するのに役立ちます。

次の図は、評価プロセスの概要を示しています。



評価オプション

AWS で Microsoft ワークロードの 2 つの AWS OLA オプションから選択できます。

- Lite バージョン – このユースケースでは、すべてのワークロードが VMware 上にあります。AWS には、RVTools からの出力を提供できます。その後、AWS は 1～5 日のターンアラウンドタイムを提供できます。このアプローチでは、VMware vCenter から直接プルしたポイントインタイム情報を使用して、サイズ設定の推奨事項を作成し、オンデマンドの料金オプションを提供します。
- フルバージョン – このユースケースでは、さまざまなクラウドプロバイダー、物理サーバー、仮想サーバーで混合環境が実行されています。AWS は、オペレーティングシステムエージェントを使用して 14～30 日間の使用状況データを収集します。これにより、AWS は、アプリケーションの使用パターンに基づいて、情報に基づいたインスタンスのサイズ決定を行うことができます。AWS は、Cloudamize などのいくつかのサードパーティー製ツールを使用して分析を完了します。AWS は AWS Partner Network と連携して、料金モデルとさまざまなアーキテクチャを考慮した複数の料金オプションを使用して、最終的な総保有コスト (TCO) 評価の提供を支援します。

完全な評価

完全な AWS OLA 評価は、1 時間の電話によって開始されます。このコール中、AWS は、移行をサポートする最適な AWS インフラストラクチャを決定し、データ収集方法を選択し、完了のタイムラインを確立するのに役立ちます。組織に検出ツールを実装するかどうかは、データ収集方法、組織のサイズ、組織がサーバーのフリートを管理するために使用するツールによって異なります。通常、使用状況データの収集には 2 週間かかります。

完全な AWS OLA プロセスには 30～45 日かかり、以下のフェーズで構成されます。

- ワークロードの範囲
- データの収集
- データを分析する
- 次のステップの計画

ワークロードの範囲

まず、AWS はユーザーとチームと協力して、評価の範囲を決定します。これは通常、環境タイプ (非本番や本番など) 別に分類されます。範囲には、ワークロードの場所が含まれます。これは、AWS に移行するワークロード、既に AWS で実行されているワークロード (Amazon EC2 の AWS OLA など)、他のクラウドプロバイダーで実行されているワークロードなどです。

データの収集

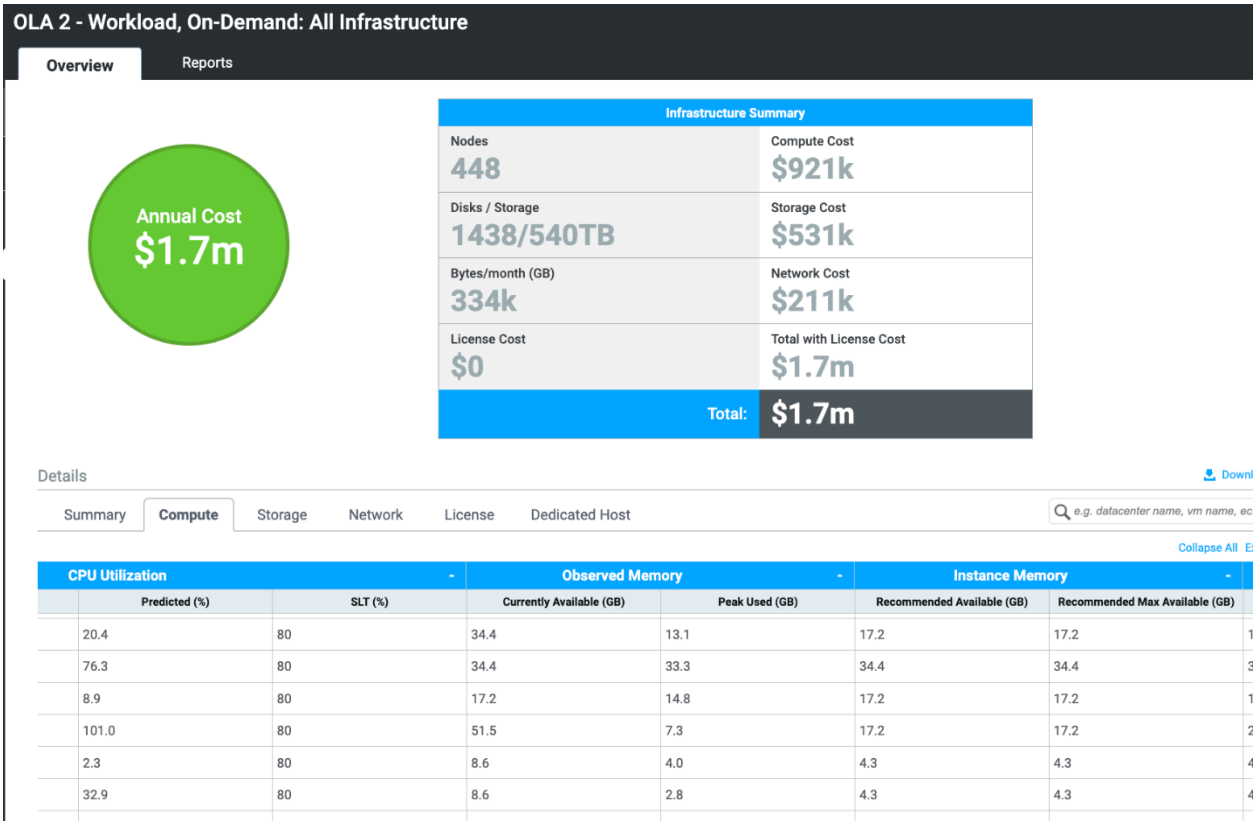
次に、AWS は、リソースの検出とサーバーからのパフォーマンスデータの収集に役立つツールをデプロイします。このツールには、次の 4 つのデプロイオプションがあります。

- ハイパーバイザーをクエリできるツール (VMware vCenter または Hyper-V 認証情報のみが必要)
- 物理マシンまたは仮想マシンにデプロイできるエージェント
- 環境とオペレーティングシステムに応じて SSH、Windows Remote Management (WinRM)、または Windows Management Instrumentation (WMI) を使用したエージェントレス検出
- フラットファイルデータ収集と分析

ツールのデプロイでは、各オプションをうまく組み合わせて、結果を統合できます。どのオプションを選ぶにしても、IT リソースに負荷をかけないようにすることがきわめて重要です。AWS は、評価プロセスができる限りすぐに利用可能になるよう努めています。セットアップを支援する簡単な電話以外にも、AWS OLA チームおよび Microsoft スペシャリストソリューションアーキテクトは、総保有コスト (TCO) 分析とレビューのための推奨事項を準備しています。

CPU 使用率、RAM 使用率、ストレージスループット、IOPS、ネットワークスループットを分析する場合、データ収集には通常 2～3 週間かかります。理想的には、この収集は、営業月のピーク時 (月末の財務報告時など) に行われます。AWS は、ピーク使用量をキャプチャしたいと考えます。これにより、適切なサイズの AWS インスタンスはどうあるべきかに関する適切な統計サンプルを提供しながら、オンプレミスで利用可能なものをパフォーマンスが超えることを保証するからです。AWS は、使用率メトリクスとさまざまなプロセッサ世代のパフォーマンスヒューリスティックを統合して、特定のワークロードに必要な CPU と RAM の正確な量をターゲットにします。これらのターゲットは通常、オンプレミスで割り当てられているものよりも小さくなります。これにより、インスタンスのサイズに関するコンピューティングコストが削減されるだけでなく、ライセンスコストも最適化されます。

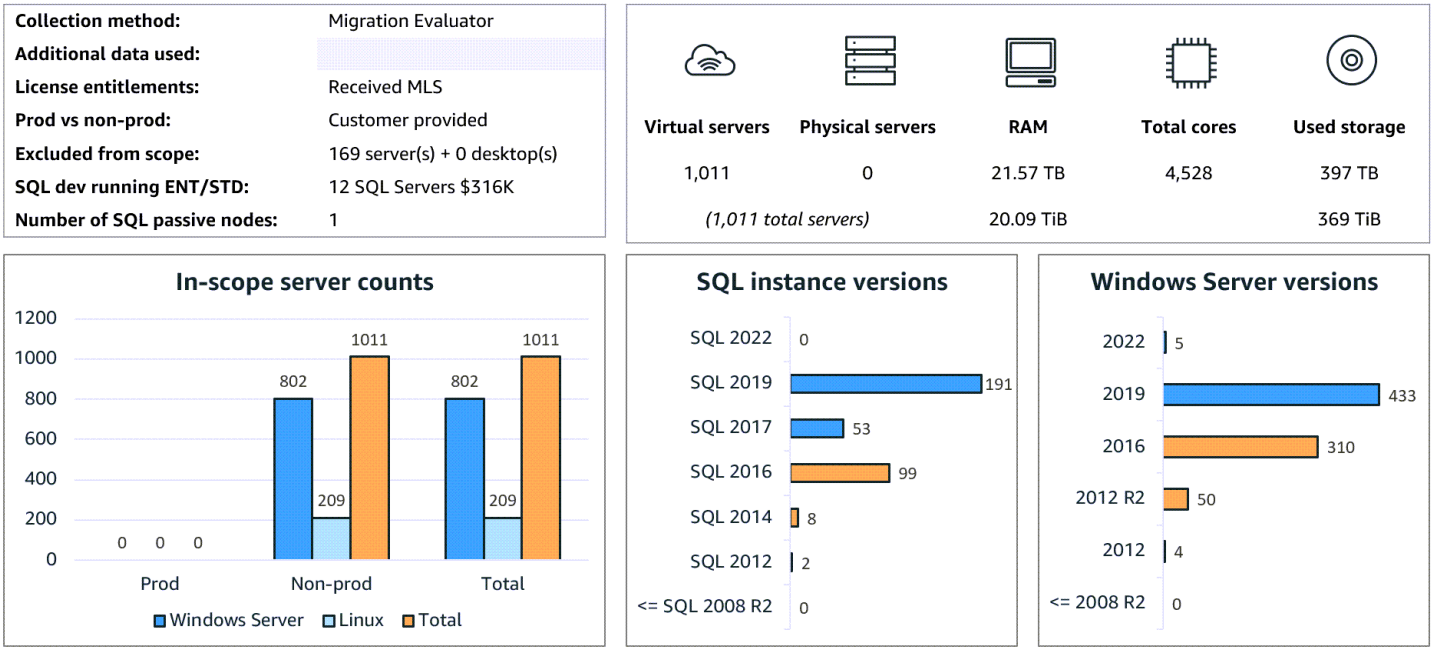
次のダッシュボードビューは、評価でキャプチャできるインフラストラクチャコストの例を示しています。



データを分析する

AWS は、データ収集の完了後に報告プレゼンテーションを提供します。AWS は、データを確認し、結果を要約してから、オンプレミスの使用状況とクラウド移行に関する推奨事項を作成します。統合の機会、伸縮性の向上 (ワークロードをオフまたは季節調整できる)、適切な SKU の機会 (例えば、SQL Server Enterprise Edition を使用中だが、リソース要件と機能の使用状況から SQL Server Standard Edition が適していることがわかるなど) を調べることで、コンピューティングとライセンスのコストを削減できます。コアによってライセンスされている SQL Server などの製品では、より高価なコンピューティングインスタンスにワークロードを配置することが経済的に理にかなっていることがよくあります。つまり、CPU プロファイルと RAM と vCPU の比率が、ライセンス込みのユースケースと Bring Your Own License (BYOL) ユースケースの両方でライセンスされたコアの数を減らす実質的な効果をもたらす場合です。

以下は、評価によって収集されたデータに基づく分析の例です。



一般的な最適化シナリオには、AWS リソース最適化の機会とサードパーティーライセンスの節約の両方を特定することが含まれます。

AWS リソース最適化の機会の例:

- ピーク使用量のオーバープロビジョニングを回避する。
- リソースの過剰な指定や過少使用を回避する。
- インスタンスのサイズを適切に設定し、最新世代の EC2 インスタンスに移行する。
- マネージドデータベースに移行することで、運用コストを節約する。

サードパーティーライセンスの節約の例:

- 同じワークロードを実行するために必要なコアを減らす。
- 不要な SQL Server Enterprise Edition とアドオンパックを取り除く。
- ゾンビサーバーを削除し、古いハードウェアを置き換える。
- BYOL およびライセンス込みオプションを使用して、将来の商業契約を削減する。
- オープンソースおよびクラウドネイティブソリューションにモダナイズする。

次のステップの計画

最後に、AWS は、収集されたパフォーマンスデータを使用して、特定のワークロードのサイズ設定とコストを見積もります。AWS は、スコープされた環境を集約して調べ、定量的分析を行うこともできます。これは、最適なオプションがオンプレミスの更新か、AWS への移行かを判断するのに役立ちます。AWS OLA の最後に用意されている TCO 分析の概要 (次の例を参照) を使用して、クラウドエコノミクスのビジネスケースを構築できます。

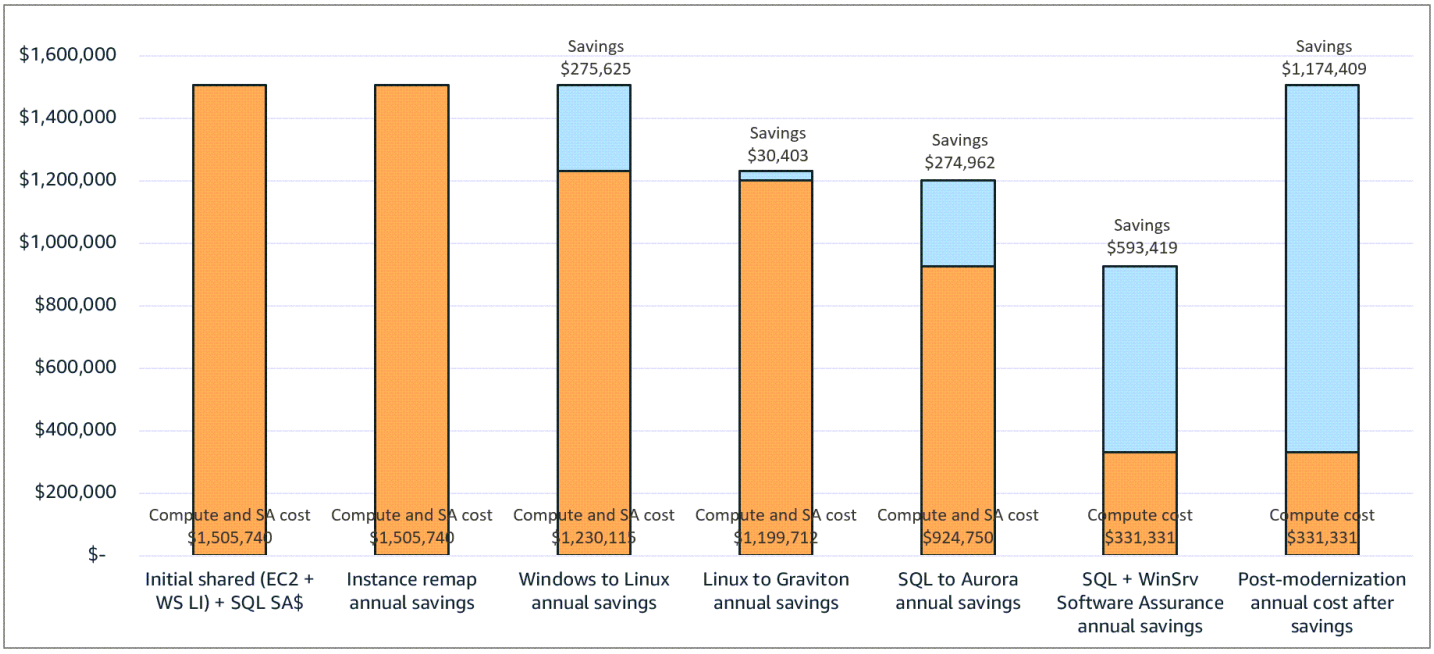
	Option 1: Amazon EC2 shared	**Option 1a: Amazon EC2 shared + power management**	**Option 2: Amazon EC2 mixed**	**Option 2a: Amazon EC2 mixed + power management**
Option details: compute	*100% Reserved Instances (RIs)*	*RIs + on-demand power management*	*100% Ris*	*RIs + on-demand power management*
Option details: Microsoft licenses	*WS LI and SQL BYOL*	*WS LI and SQL BYOL*	*WS BYOL or LI+SQL BYOL*	*WS BYOL or LI+SQL BYOL*
Compute costs[1]				
Year 1 compute cost	$414,546	$482,623	$504,019	$513,941
Year 1 vendor license included cost	$392,858	$244,415	$9,804	$4,783
	$807,404	**$727,038**	**$513,823**	**$518,724**
Total compute savings in year 1, compared to Option 1	—	*10% ($80,366)*	*36% ($293,581)*	*36% ($288,680)*
Storage and networking costs[2]				
Annual estimated storage cost	$336,494	$336,494	$336,494	$336,494
Annual estimated networking cost	$41,455	$41,455	$41,455	$41,455
	$377,949	**$377,949**	**$377,949**	**$377,949**
Microsoft license costs**				
WS/CIS annual Software Assurance (SA) + current SPLA/Subs cost	$0	$0	$0	$0
WS/CIS license + SA + SPLA/Subs true-up cost	$0	$0	$0	$0
SQL annual SA + current SPLA/Subs cost	$0	$0	$0	$0
SQL license SA + current SPLA/Subs true-up cost	$0	$0	$0	$0
	$0	**$0**	**$0**	**$0**
Total estimated costs	**$1,185,353**	**$1,104,987**	**$891,772**	**$896,673**
Annual TCO savings in year 1, compared to Option 1	—	*7% ($80,366)*	*25% ($293,581)*	*24% ($288,680)*

[1] Pricing model used: 3-year, no upfront RI
[2] Software Assurance and true-up costs provided by Microsoft

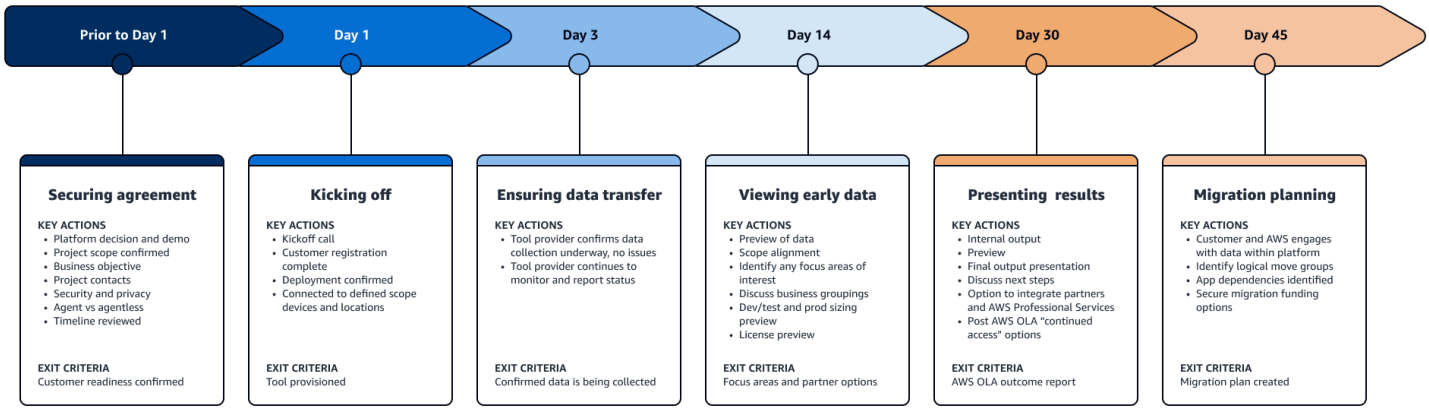
AWS OLA は、次のような提案を行うことで、モダナイズが既存のワークロードに与える影響についてのインサイトも提供します。

- Linux オペレーティングシステムに移動する。
- ARM プロセッサ (AWS Graviton) のアプリケーションサポートを追加する。
- SQL Server ワークロードを Amazon Aurora に移動する。
- Windows および SQL Server ワークロードをオープンソーステクノロジーに移動することで、ソフトウェア保証を削除する。

次の図は、Windows から Linux への移動や SQL Server から Aurora への移動などのモダナイズ手法によって実現できるコスト削減を示しています。



完全な AWS OLA プロセスは、開始から終了まで約 45 日かかります。次の図は、タイムラインの例を示しています。



純粋な VMware 環境があり、RVTools からの出力を提供できる場合は、このタイムラインを 1 営業週に短縮できます。さらに、AWS は、CPU 平均、CPU ピーク、RAM 平均、RAM ピークなどのアセットと使用率データを含むフラットファイルを分析できます。

評価の影響

平均的なお客様は、通常、サイズ適正化作業により 20～30% のコスト削減を経験します。サイズ適正化は、使用状況データに基づいてソースワークロードを最適なサイズの AWS インスタンスに一致させます。これらのサイズ適正化調整は、AWS 環境の月額コストを削減するだけでなく、多くの場合、組織内の他の場所で節約につながります。例えば、Windows または SQL Server のライセンス

を 20～30% 増やすと、Microsoft との次の調整を減らしたり、追加の基幹業務アプリケーションのライセンスを解放したりできます。SQL Server ワークロードの統合とサイズ適正化は、一般的に最も劇的な財務上の利益を実現する場所です。

AWS は、システムをモダナイズバケットに分類するのに役立ちます。一部のシステムはレガシーであり、財務的には触れることができませんが、最も大きな節約が実現されるコンテナやサーバーレスアプリケーションにモダナイズされるシステムもあります。AWS チームとの会話は、クラウドが何を可能にするかについての一般化されたトピックから、特定のワークロードをモダナイズする方法と理由についてのより具体的な議論に移行します。また、AWS は、潜在的なイノベーションの機会の探索にも役立ちます。

次のステップ

オンプレミス環境または AWS で実行されている Microsoft ワークロードのコスト最適化ジャーニーを開始する場合は、AWS アカウントチームと連携して AWS OLA をリクエストしてください。AWS チームメンバーは、質問に回答し、AWS OLA が最終的にユーザーと組織にとって適切な選択であるかどうかを決定するのを支援します。または、AWS OLA をオンラインでリクエストすることもできます。

その他のリソース

- AWS Optimization and Licensing Assessment (AWS ドキュメント)
- 「AWS re:Invent 2022 - How to save costs and optimize Microsoft workloads on AWS (ENT205)」(YouTube)

Amazon EC2 の Windows

Amazon Elastic Compute Cloud (Amazon EC2) は、Windows ワークロードの実行に最適な、柔軟性とスケーラビリティに優れたクラウドコンピューティングプラットフォームです。Amazon EC2 を使用して、 AWS クラウドの安全で信頼性と可用性が高く、順応性のあるインフラストラクチャで Windows Server ワークロードをデプロイ、管理、およびスケールすることができます。Amazon EC2 で Windows ワークロードを実行する場合の主な利点を以下に示します。

- スケーラビリティ – Amazon EC2 を使用すると、Windows ワークロードを簡単にスケールして、変化する要件に対応できます。需要の増加に対応するために新しい EC2 インスタンスをすばやく作成し、不要になったインスタンスを簡単に終了できます。お客様は、実際に使用したリソースに対してのみ料金を支払います。
- 柔軟性 – Amazon EC2 上の Windows は、汎用インスタンスからメモリやコンピューティング最適化インスタンスまで、さまざまなワークロード要件を満たすように設計された幅広いインスタンスタイプをサポートしています。この柔軟性により、特定の Windows ベースのアプリケーションに最適なインスタンスタイプを選択し、パフォーマンスを最大化してコストを最小限に抑えることができます。
- セキュリティ – ネットワークファイアウォール、データ暗号化、安全なアクセスコントロールなど、Windows ワークロードに複数のセキュリティレイヤー AWS を提供します。つまり、セキュリティ設定および構成を完全に制御しながら、アプリケーションとデータが保護されていることを信頼できます。
- コスト効率 – 従量制料金モデルを使用すると、使用したリソースに対してのみ料金を支払うことができるため、ハードウェアやソフトウェアに先行投資する必要がなくなります。また、このモデルにより、コストの最適化、設備投資の削減、運用効率の向上が可能になります。これは、あらゆる規模のビジネスに最適な料金モデルです。

本ガイドのこのセクションでは、次のトピックについて説明します。

- 停止スケジュールと開始スケジュールを自動化する
- Windows ワークロードを適切なサイズに設定する
- Windows ワークロードに適したインスタンスタイプを選択する
- Windows および SQL Server ワークロードのライセンスを持ち込む
- Amazon EC2 での Windows の支出を最適化する
- AWS ツールを使用したコストのモニタリング

停止スケジュールと開始スケジュールを自動化する

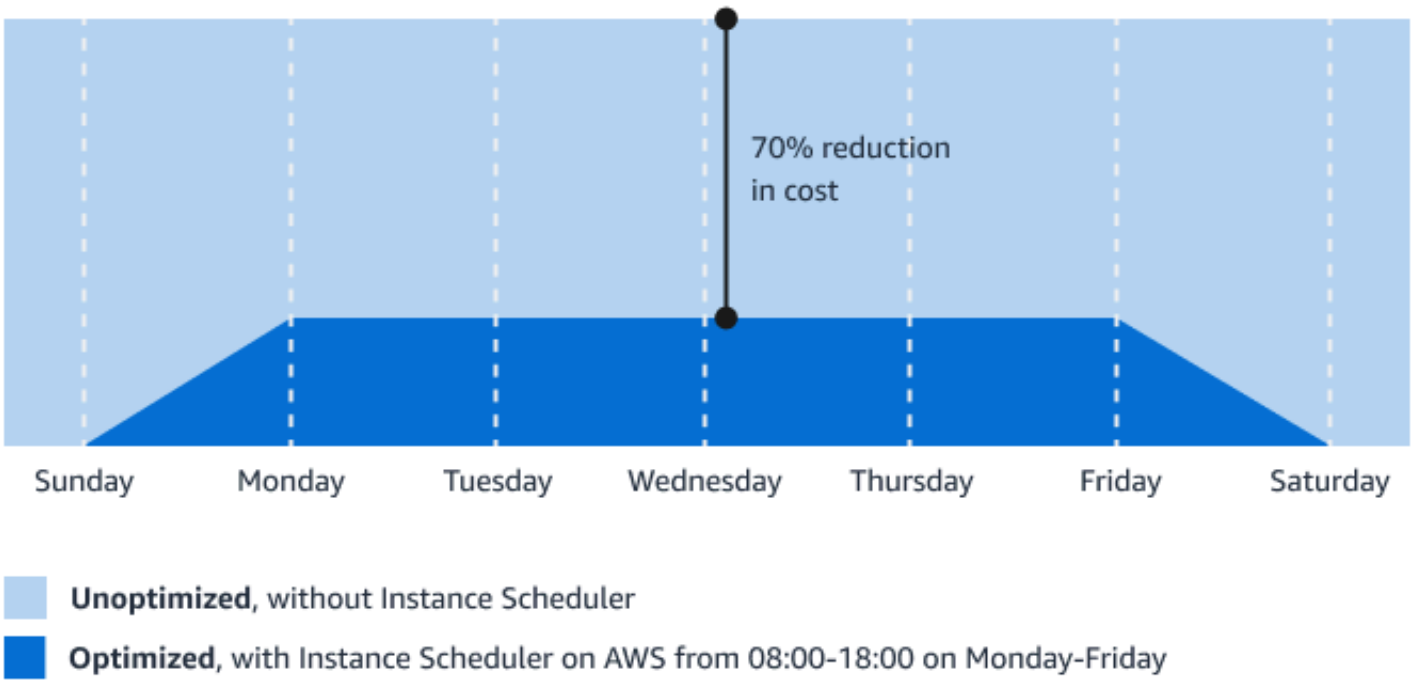
概要:

AWSの Instance Scheduler は、Amazon EC2 および Amazon Relational Database Service (Amazon RDS) インスタンスの開始と停止を自動化することで、運用コストを削減するのを支援します。すべてのインスタンスをフル使用で継続的に実行したままにすると、使用されていないリソースに対して料金が発生する可能性があります。インスタンススケジューラをオンに AWS すると、営業時間外、週末、または使用量が少ないその他の期間など、不要な時間帯にインスタンスをオフにできます。これにより、時間の経過とともに大幅なコスト削減につながる可能性があります。

の Instance Scheduler は、クロスアカウントインスタンススケジューリング、自動タグ付け、コマンドラインインターフェイスまたはAWS Systems Managerメンテナンスウィンドウを使用してスケジュールまたは期間を設定する機能 AWS も提供します。これらの機能により、インスタンスをより効果的に管理し、さまざまなプロジェクトやチームにまたがるコストを正確に追跡および割り当てることができます。

ケーススタディ

実稼働環境で AWS で Instance Scheduler を使用して、毎日営業時間外にインスタンスを自動的に停止する会社の例を考えてみましょう。この会社がすべてのインスタンスをフル使用で実行したままにしている場合、通常の営業時間内にのみ必要なインスタンスに対して最大 70% のコスト削減を実現できます。次のグラフは、週の使用時間が 168 時間から 50 時間に削減された場合を示しています。

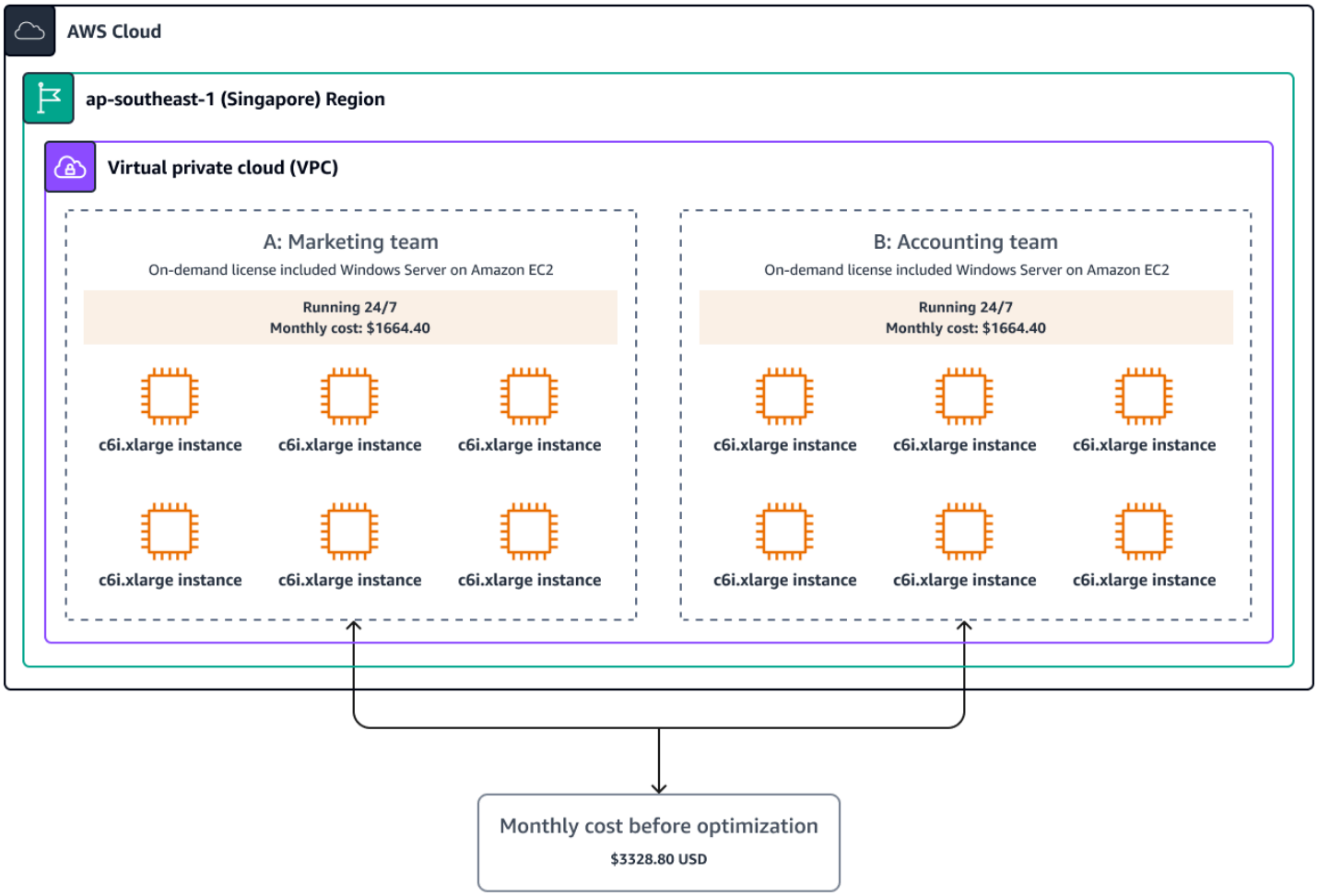


別の例を考えてみましょう。電力会社である Jamaica Public Service Company Limited (JPS) は、データベースを Amazon RDS に移行しました。現在、JPS は Amazon EC2 を使用して API サービスをホストし、他のアプリケーションを実行しています。JPS では、 の Instance Scheduler が非本番環境を管理するための主要なツール AWS になりました。JPS は で Instance Scheduler を使用して AWS 開発コストを削減し、チームのニーズと作業スケジュールに基づいて EC2 インスタンスを管理しました。これにより、JPS はコストを 40% 削減できました。詳細については、「Jamaica Public Service Migrates Efficiently to the Cloud, Reduces Costs by 40% Using AWS Instance Scheduler」の AWS ケーススタディを参照してください。

コスト最適化シナリオ

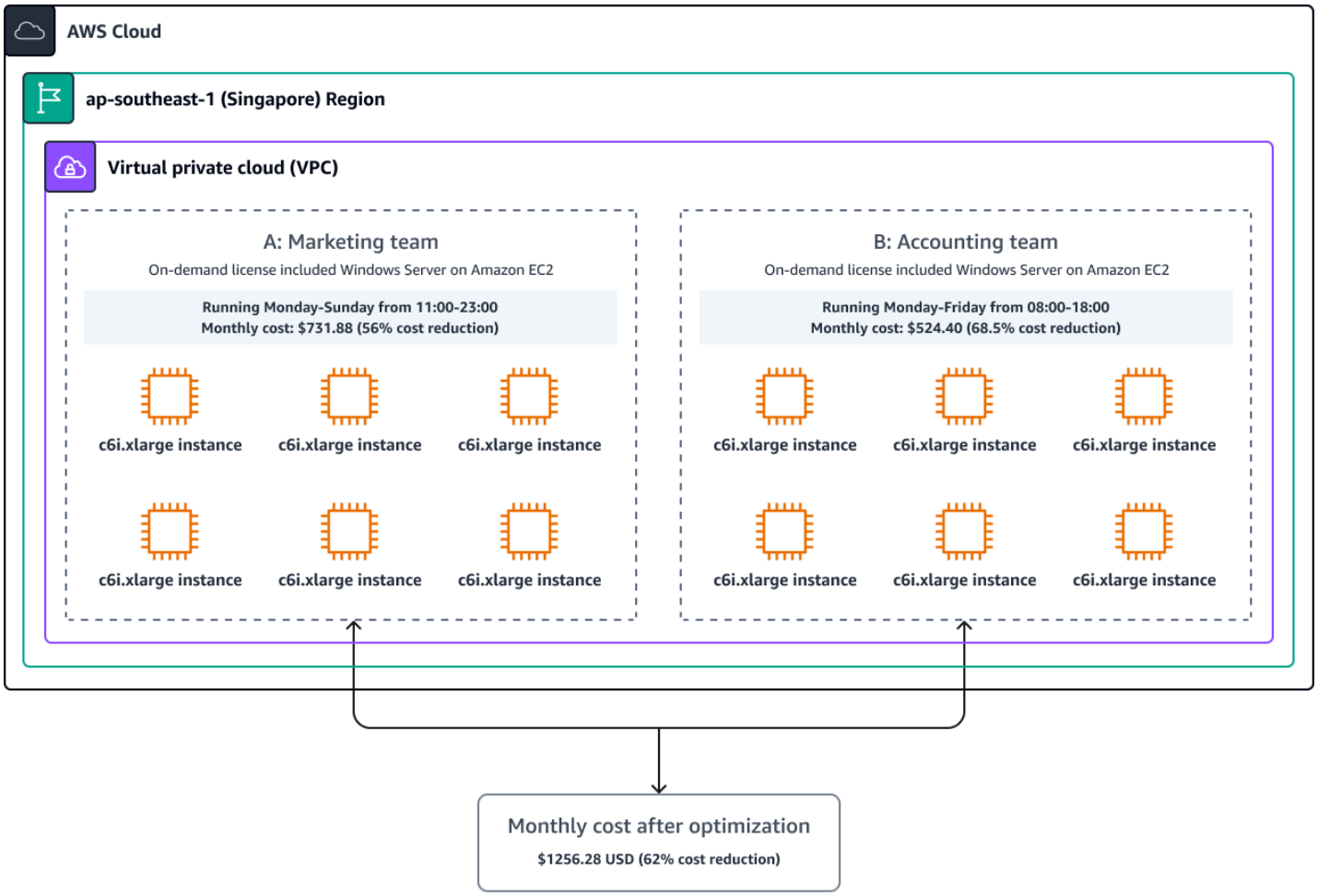
次のシナリオ例は、 で Instance Scheduler を使用するコスト上の利点を説明するのに役立ちます AWS。このシナリオでは、シンガポールの大手小売企業が Amazon EC2 に 2 つの Windows 環境をデプロイしています。ワークロード A と呼ばれる最初の環境は、店舗の営業中にマーケティングチームが店舗内の取引をリアルタイムで分析するために使用されます。ワークロード B と呼ばれる 2 番目の環境は、通常の営業時間内にのみ作業する経理チーム用に予約されています。両方の環境の現在の運用スケジュール (24 時間週 7 日) は、現在の使用パターンを考慮すると理想的ではなく、会社の運用コストを削減するために最適化が必要です。

次の図は、最適化前の月額コストを示しています。



例えば、3 月は 31 日あり、そのうちの 23 日は平日です。マーケティングチームが で Instance Scheduler を使用し AWS 、必要な場合にのみインスタンスを操作する場合 (つまり、1 か月あたり 730 時間ではなく 1 か月あたり 321 時間) 、毎月 932.52 USD を節約できる可能性があります。これにより、運用コストが 56% 削減されます。経理チームにも大きな利点があり、インスタンスの使用時間が 1 か月あたり 730 時間から 230 時間に削減されます。これにより、1,140 USD、つまり 68.5% の削減になります。同社は、合計で 1 か月あたり 2,072.52 USD (62% の削減に相当)、つまり年間 24,870.24 USD を節約できます。

次の図は、最適化後の月額コストを示しています。



> **Note**
>
> この例の料金は、2023 年 3 月に AWS 料金見積りツール を使用して算出されました。

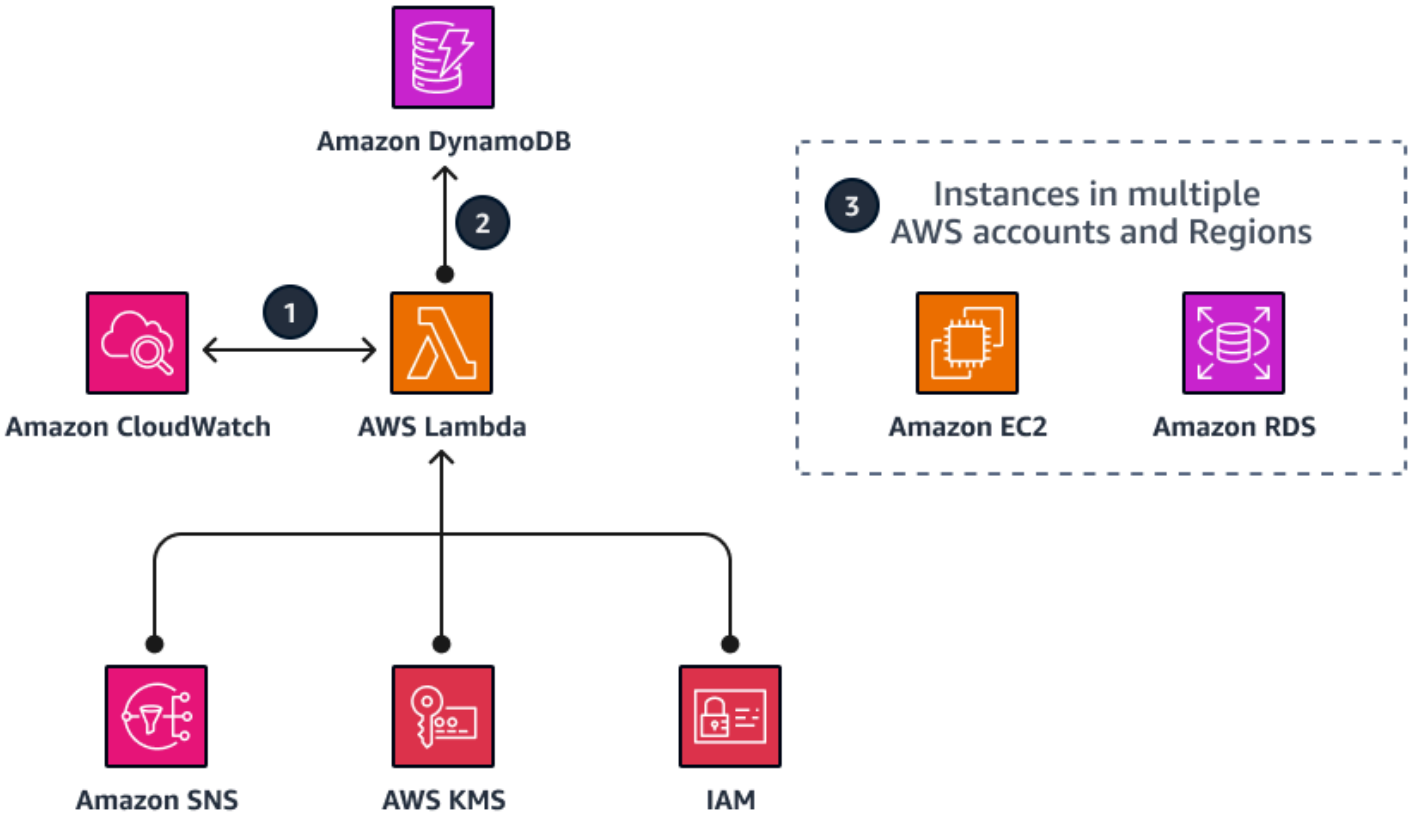
コスト最適化の推奨事項

このセクションでは、前の「コスト最適化シナリオ」セクションで説明したシナリオ例に基づいて、 AWS の Instance Scheduler をデプロイおよび設定する方法について説明します。Instance Scheduler を使用してコストを最適化するには、次のステップを実行することをお勧めします AWS。

1. Instance Scheduler スタックを起動する
2. 期間を設定する
3. スケジュールを設定する

4. インスタンスにタグ付けする

次のアーキテクチャ図は、Instance Scheduler スタック AWS クラウド によって に作成される内容を示しています。



この図は、次のワークフローステップを示しています。

1. AWS CloudFormation テンプレートは、定義した間隔で Amazon CloudWatch イベントを設定します。このイベントは AWS Lambda 関数を呼び出します。設定中に、 アカウント AWS リージョン と アカウントを定義します。また、 の Instance Scheduler が、該当する Amazon EC2 インスタンス、Amazon RDS インスタンス、およびクラスターにスケジュールを関連付けるために AWS 使用するカスタムタグを定義します。
2. スケジュール設定値は Amazon DynamoDB に保存され、Lambda 関数は実行されるたびにそれらを取得します。その後、該当するインスタンスにカスタムタグを適用できます。
3. Instance Scheduler の初期設定時に、該当する Amazon EC2 および Amazon RDS インスタンスを識別するためのタグキーを定義します。スケジュールを作成するときに指定した名前は、タグ付けされたリソースに適用するスケジュールを識別するタグ値として使用されます。

Instance Scheduler スタックを起動する

このセクションでは、 AWSの Instance Scheduler の CloudFormation スタックを起動する方法について説明します。

> **Note**
>
> Instance Scheduler の実行中に AWS のサービス 使用した のコストは、お客様の負担となります AWS。2023 年 1 月現在、us-east-1 リージョンのデフォルト設定でこのソリューションを実行する場合のコストは、Lambda 料金で 1 か月あたり約 9.90 USD です。Lambda 無料利用枠の月間使用クレジットがある場合はそれ以下になります。詳細については、「 AWS ソリューションライブラリ」の「Instance Scheduler on AWS Implementation Guide」の「Cost」セクションを参照してください。

Instance Scheduler スタックを起動するには、次のステップを実行します。

1. AWS マネジメントコンソール にサインインし、[Launch Solution] (ダウンロード可能なテンプレート) を選択して `instance-scheduler-on-aws.template` CloudFormation テンプレートを起動します。

   > **Note**
   >
   > 実装の開始点としてテンプレートをダウンロードすることもできます。

2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別のリージョンで Instance Scheduler を起動するには、コンソールナビゲーションバーのリージョンセレクターを使用します。

   > **Note**
   >
   > この例では、アジアパシフィック (シンガポール) リージョンを使用しています。

3. [スタックの作成] ページの [前提条件 - テンプレートの準備] セクションで、[テンプレートの準備完了] オプションが選択されていることを確認します。[テンプレートソース] セクションで、[Amazon S3 URL] オプションが選択されていることを確認します。
4. 正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。

5. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前文字の制限については、 AWS Identity and Access Management (IAM) ドキュメントの「IAM および STS の制限 」を参照してください。このガイドの例のスタック名は `MyInstanceScheduler` です。

> **Note**
>
> スタック名は 28 文字を超えることはできません。

6. [パラメータ] で、テンプレートのパラメータを確認し、必要に応じて変更します。
7. [次へ] を選択します。[スタックオプションの設定] ページで、[次へ] を選択します。
8. [レビュー] ページで、設定を確認します。テンプレートが IAM リソースを作成することを確認するチェックボックスを選択します。
9. [作成] を選択してスタックをデプロイします。

期間を設定する

CloudFormation テンプレートをデプロイすると、ソリューションによって、独自のカスタム期間ルールおよびスケジュールを作成するための参照として使用できる、サンプル期間ルールおよびスケジュールが含まれる DynamoDB テーブルが作成されます。期間設定の例については、「 AWS での Instance Scheduler」ドキュメントの「サンプルスケジュール」を参照してください。

このシナリオのステップを完了するには、各ワークロードに対応し、特定のニーズを満たす期間を生成する必要があります。例えば、次のようになります。

```
Period 1 (Workload A):
    Name: retail-hours
    Days: Monday to Sunday
    Hours: 1100 - 2300
Period 2 (Workload B):
    Name: office-hours
    Days: Monday to Friday
    Hours: 0800 - 1800
```

期間を設定するには、次のステップを実行します。

1. DynamoDB コンソールにサインインし、 AWSの Instance Scheduler の CloudFormation テンプレートを起動したリージョンと同じリージョンにいることを確認します。

2. ナビゲーションペインで [テーブル] を選択して、ConfigTable という名前のテーブルを選択します。
3. [テーブルアイテムの探索] を選択します。
4. 営業時間の期間を作成するには、office-hours 項目の period を選択します。
5. [項目の編集] ページで、begintime の値を 0800 に、endtime の値を 1800 に変更します。weekdays はデフォルト値のままにしておきます。

> **Note**
>
> begintime と endtime の値は、インスタンスを起動および停止するタイミングを決定し、weekdays の値は、このスケジュールが適用される曜日 (この例では月曜日から金曜日) を決定します。

6. [Save changes] (変更の保存) をクリックします。
7. office-hours の期間を複製し、それを使用して小売時間の新しい期間を作成するには、office-hours 項目の period を選択します。次に、[アクション] メニューから [項目の複製] を選択します。
8. ニーズに合わせて属性を変更します。次の属性は、シナリオ例の要件を満たすために使用されます。

```
type: period
name: retail-hours
begintime: 11:00
description: Retail hours
endtime: 23:00
weekdays: mon-sun
```

9. [項目を作成] を選択します。
10. DynamoDB ConfigTable で、項目リストに表示されている、先ほど作成した 2 つの期間を特定します。

スケジュールを設定する

での Instance Scheduler のコンテキストでは AWS、スケジュールは 1 つ以上の期間および関連するタイムゾーンの適用を指します。これらのスケジュールは、タグとしてインスタンスに割り当てられます。このセクションでは、2 つのサンプルワークロードのさまざまな時間パターンに対応するために 2 つのスケジュール (以下を参照) を作成し、そのスケジュールを前のセクションで作成した期間に関連付ける方法を示します。

```
Schedule 1:
    Name: singapore-office-hours
    Period: office-hours
    Timezone: Asia/Singapore
Schedule 2:
    Name: singapore-retail-hours
    Period: retail-hours
    Timezone: Asia/Singapore
```

スケジュールを作成して設定するには、次のステップを実行します。

1. DynamoDB コンソールにサインインし、 AWSの Instance Scheduler の CloudFormation テンプレートを起動したリージョンと同じリージョンにいることを確認します。
2. ナビゲーションペインで [テーブル] を選択して、ConfigTable という名前のテーブルを選択します。
3. [テーブルアイテムの探索] を選択します。
4. 英国の営業時間スケジュールを複製し、それを使用して営業時間 (この例ではシンガポールの営業時間) の新しいスケジュールを作成するには、uk-office-hours 項目の schedule を選択します。次に、[アクション] メニューから [項目の複製] を選択します。
5. ニーズに合わせて属性を変更します。次の属性は、シナリオ例の要件を満たすために使用されます。

```
type: schedule
name: singapore-office-hours
description: Office hours in Singapore
periods: office-hours
timezone: Asia/Singapore
```

6. [項目を作成] を選択します。
7. ステップ 4～6 を繰り返し、次の属性値を使用してシンガポールの小売時間のスケジュールを作成します。

```
type: schedule
name: singapore-retail-hours
description: Retail hours in Singapore
periods: retail-hours
timezone: Asia/Singapore
```

8. DynamoDB ConfigTable で、作成した 2 つのスケジュールと 2 つの期間を特定します。

インスタンスにタグ付けする

スケジュールを確立したら、タグを使用して、使用する特定のインスタンスにスケジュールを割り当てる必要があります。AWS Resource Groups 内のタグエディタを使用して、Amazon EC2 インスタンスにタグを生成して割り当てることができます。

1. AWS マネジメントコンソール にサインインし、以前に CloudFormation テンプレートを起動したリージョンと同じリージョンにいることを確認します。
2. リソースグループコンソールを開きます。ナビゲーションペインで、[タグ付け] を展開し、[タグエディタ] を選択します。
3. [タグ付けするリソースを検索] セクションの [リージョン] で、リージョンを選択します。[リソースタイプ] で、Amazon EC2 または Amazon RDS を選択します。このシナリオでは、ワークロード A の Amazon EC2 インスタンスに焦点を当てます。マーケティングチームは、シンガポールリージョンでワークロード A を使用しています。このワークロードのリソースには、既に Department キーと Marketing 値がタグ付けされています。このタグを使用してインスタンスを検索できます。
4. [リソースを検索] を選択します。
5. 検索結果のリストからスケジュールに含めるインスタンスを選択し、[選択したリソースのタグを管理する] を選択します。
6. [選択したすべてのリソースのタグを編集する] セクションで、[タグの追加] を選択して、Instance Scheduler のスケジュールタグを EC2 インスタンスに追加します。schedulea (以前に DynamoDB で作成) に一致するタグキーとタグ値を使用できます。
7. [タグキー] には Schedule を追加します。[タグ値] には singapore-retail-hours と入力します。
8. [タグの変更を確認して適用] を選択します。
9. 選択したすべての EC2 インスタンスにタグを適用するには、[選択したすべてに変更を適用する] を選択します。

10適用する追加のスケジュールについては、ステップ 3～9 を繰り返します。

検証結果

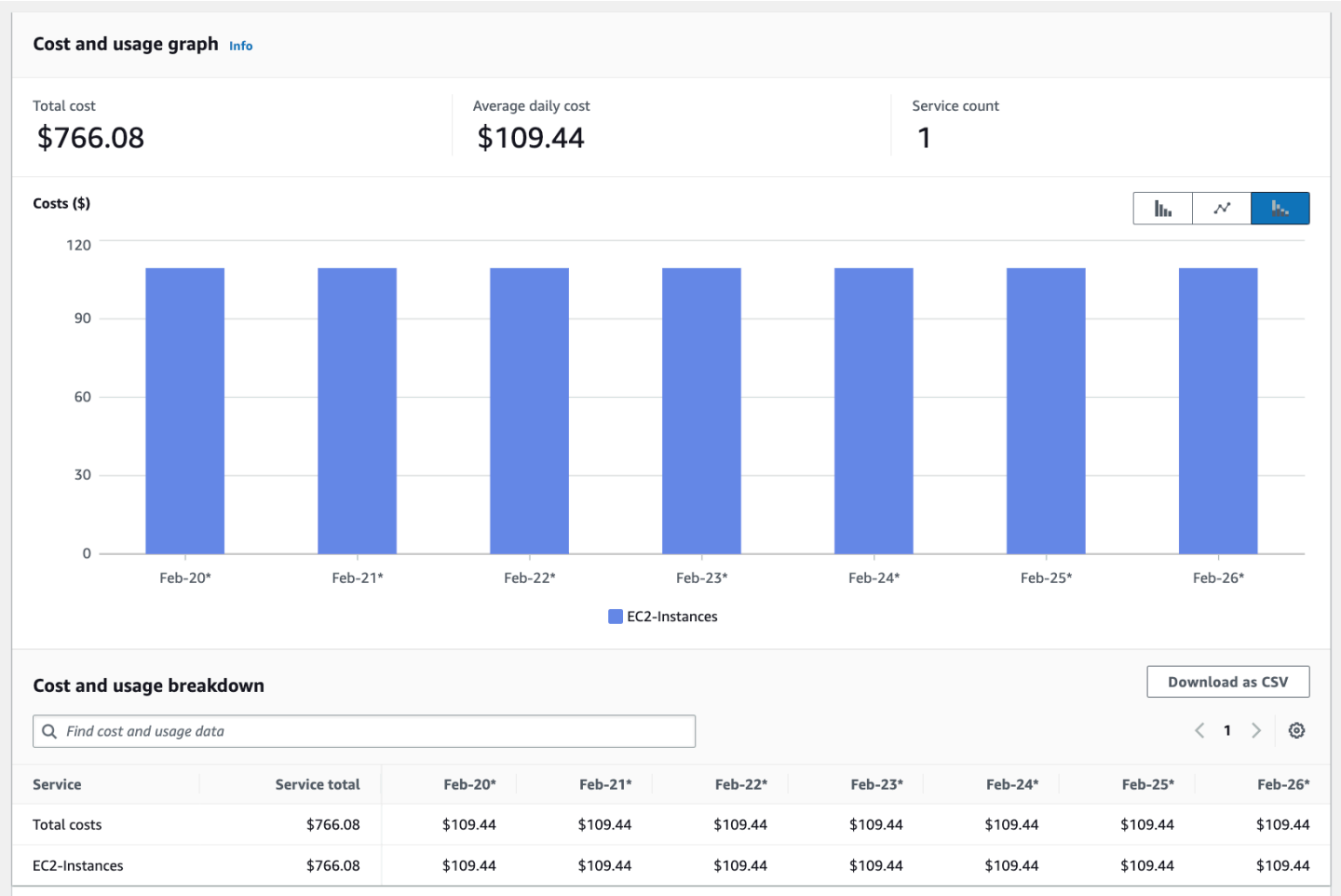
AWS Cost Explorer を使用して、 で Instance Scheduler を使用するコストメリットを測定することをお勧めします AWS。これを行うには、Cost Explorer を使用できます。

- Instance Scheduler によって管理されるインスタンスなど、EC2 インスタンスに関連するコストを表示および分析します。

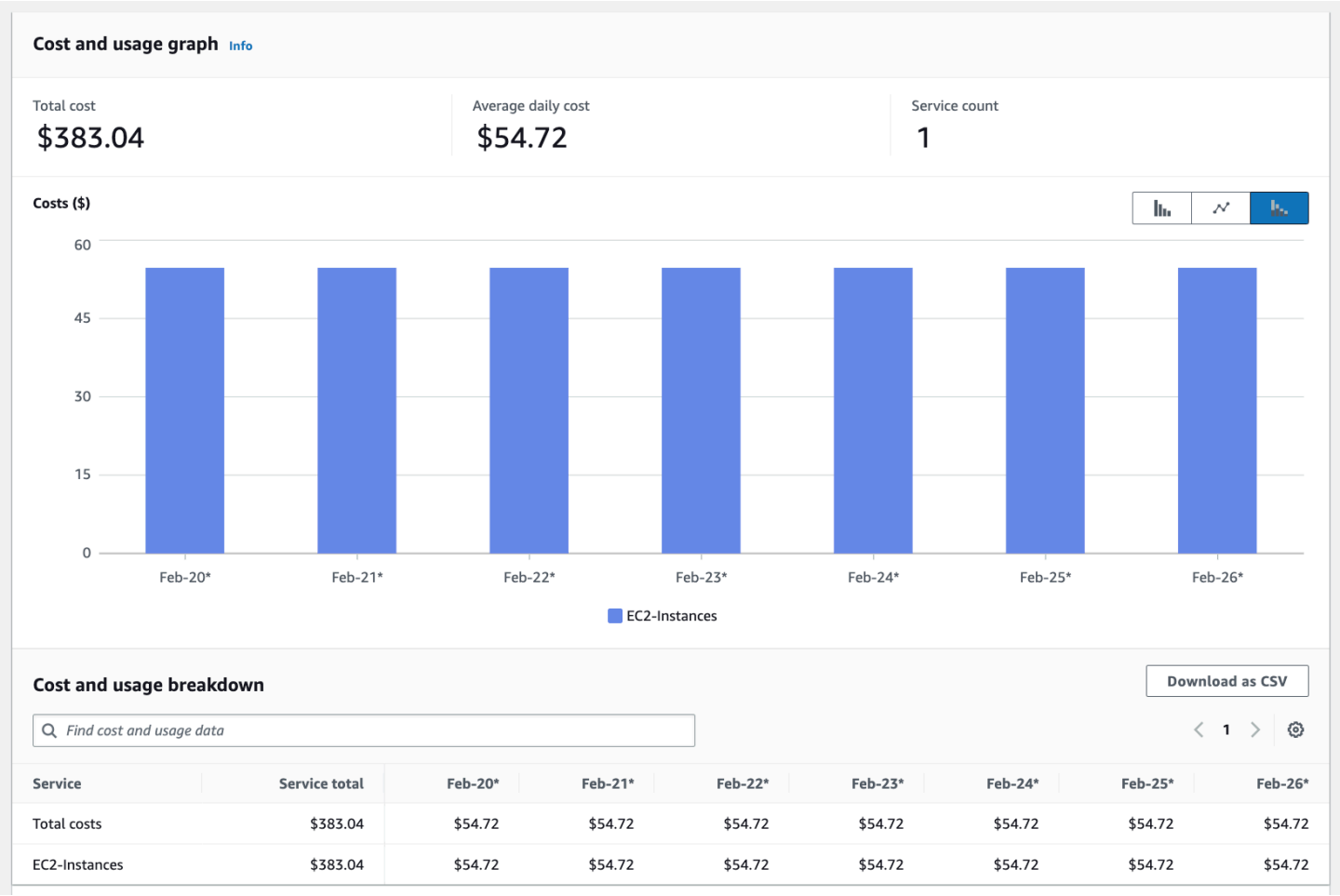
- Cost Explorer ビューをタグでフィルタリングして、特定のワークロードに焦点を当て、Instance Scheduler を使用して達成されたコスト削減を詳細に把握できるようにします。
- Instance Scheduler の使用による財務上の影響について洞察します。
- さらなるコスト最適化の機会を特定し、データ駆動型の意思決定を行って AWS 支出を最適化します。

次のグラフは、Instance Scheduler を使用して最適化する前の、7 日間 (月曜日から日曜日) のワークロード A とワークロード B の運用コストを示しています。

ワークロード A とワークロード B の合計費用

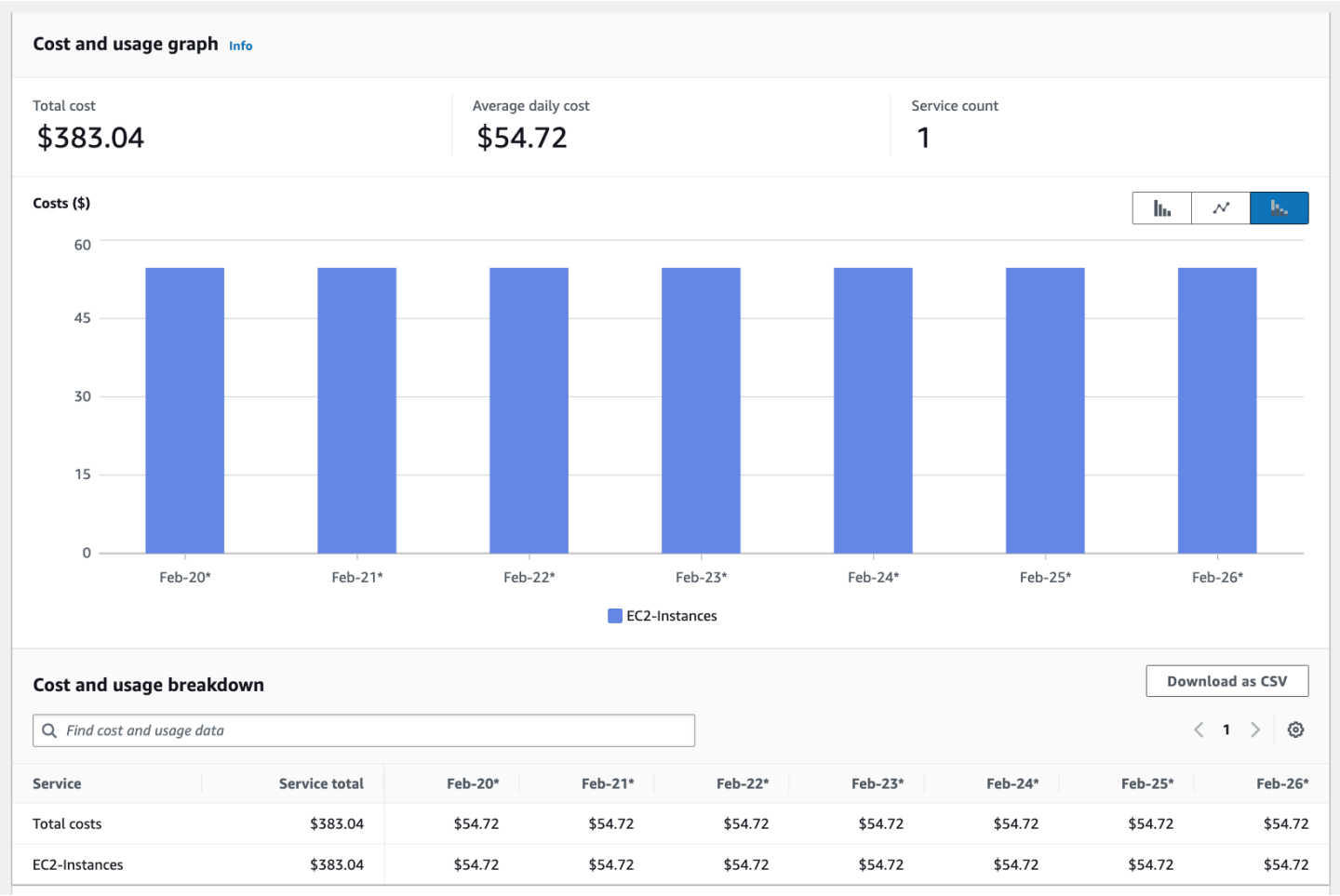


ワークロード A の費用



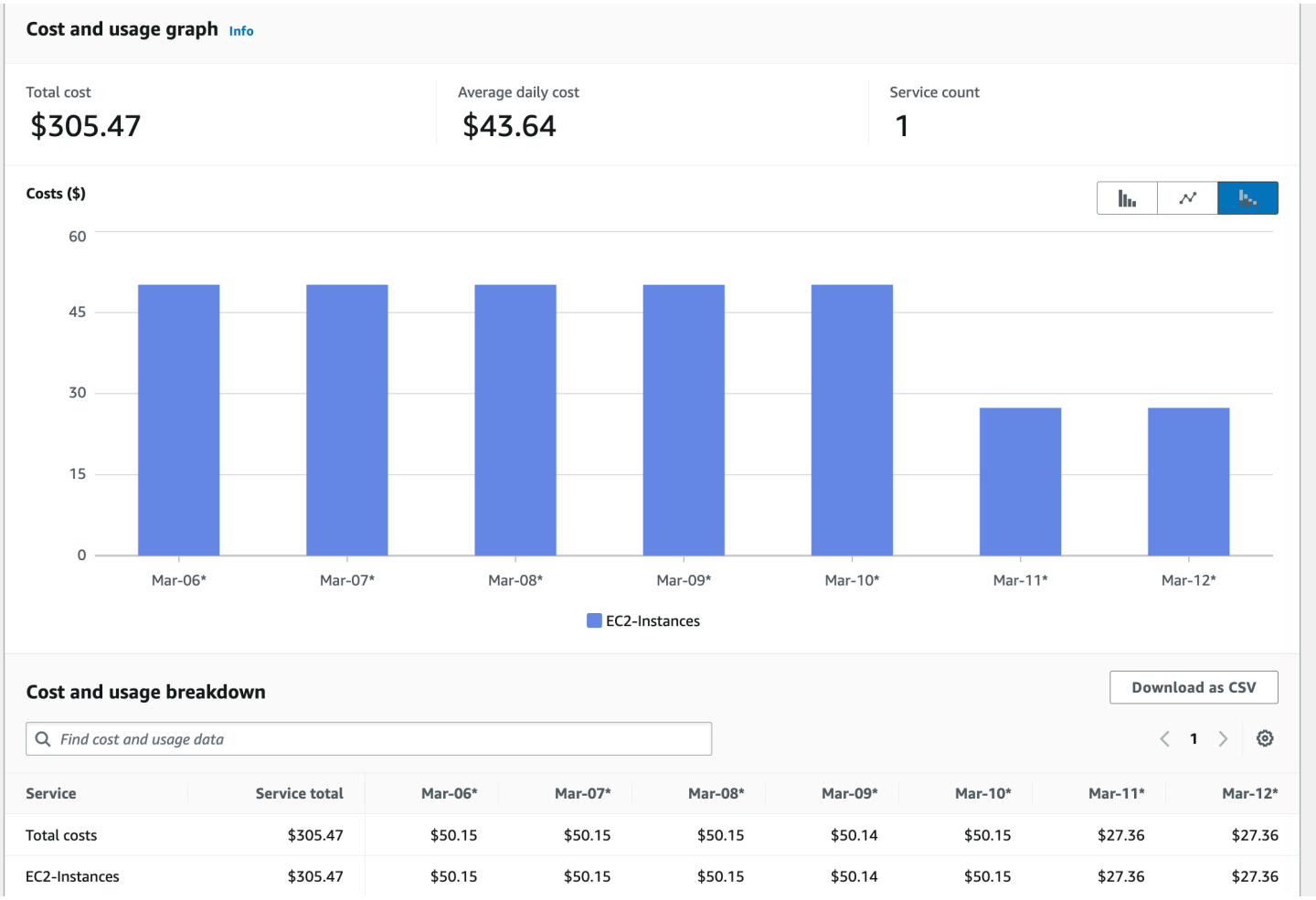
Service	Service total	Feb-20*	Feb-21*	Feb-22*	Feb-23*	Feb-24*	Feb-25*	Feb-26*
Total costs	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72
EC2-Instances	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72

ワークロード B の費用



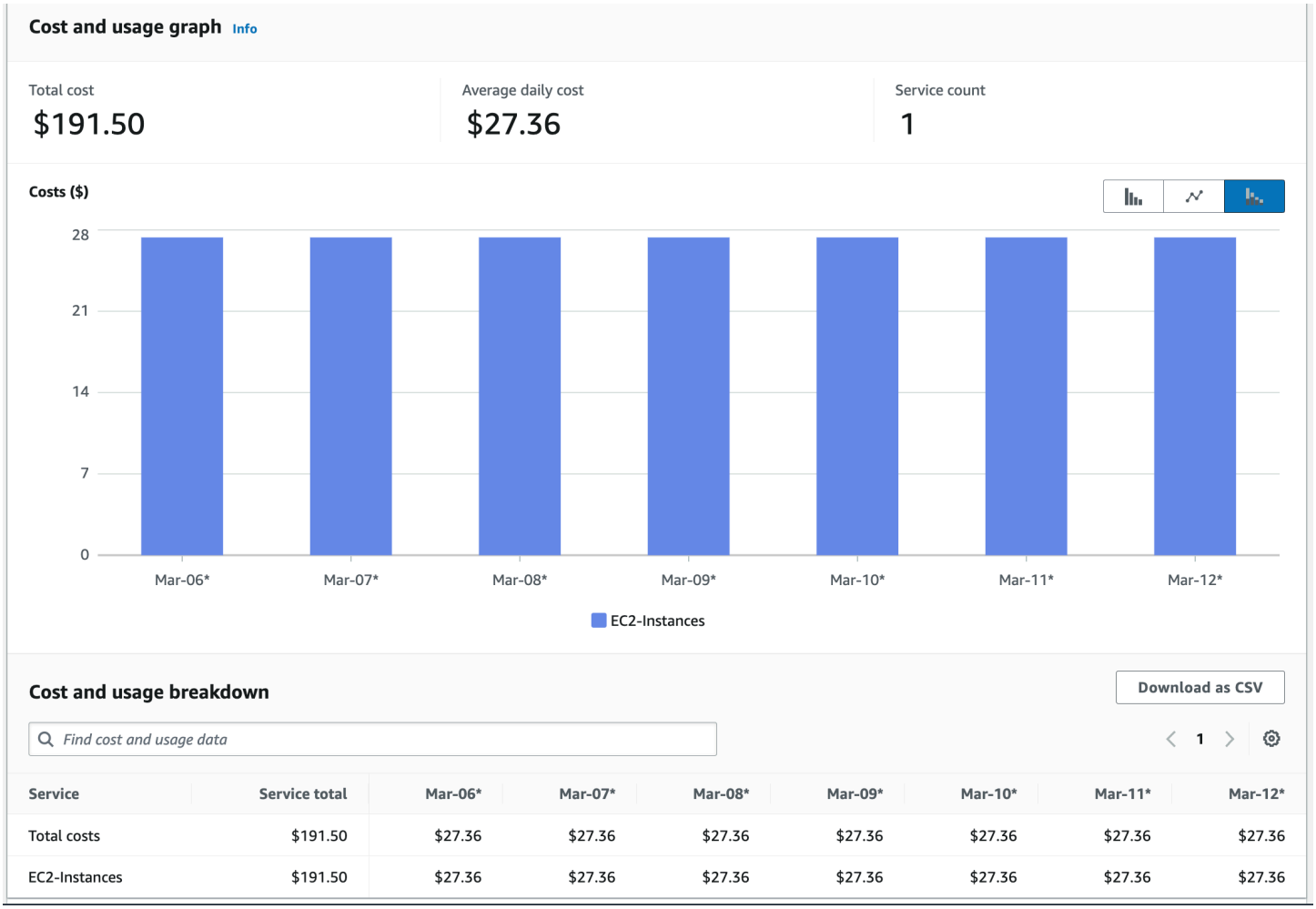
このシナリオでは、Cost Explorer は、 AWSの Instance Scheduler を実装した結果のコスト削減を示します。次のグラフは、最適化後の 7 日間 (月曜日から日曜日) のワークロード A とワークロード B の運用コストを示しています。

ワークロード A とワークロード B の合計費用

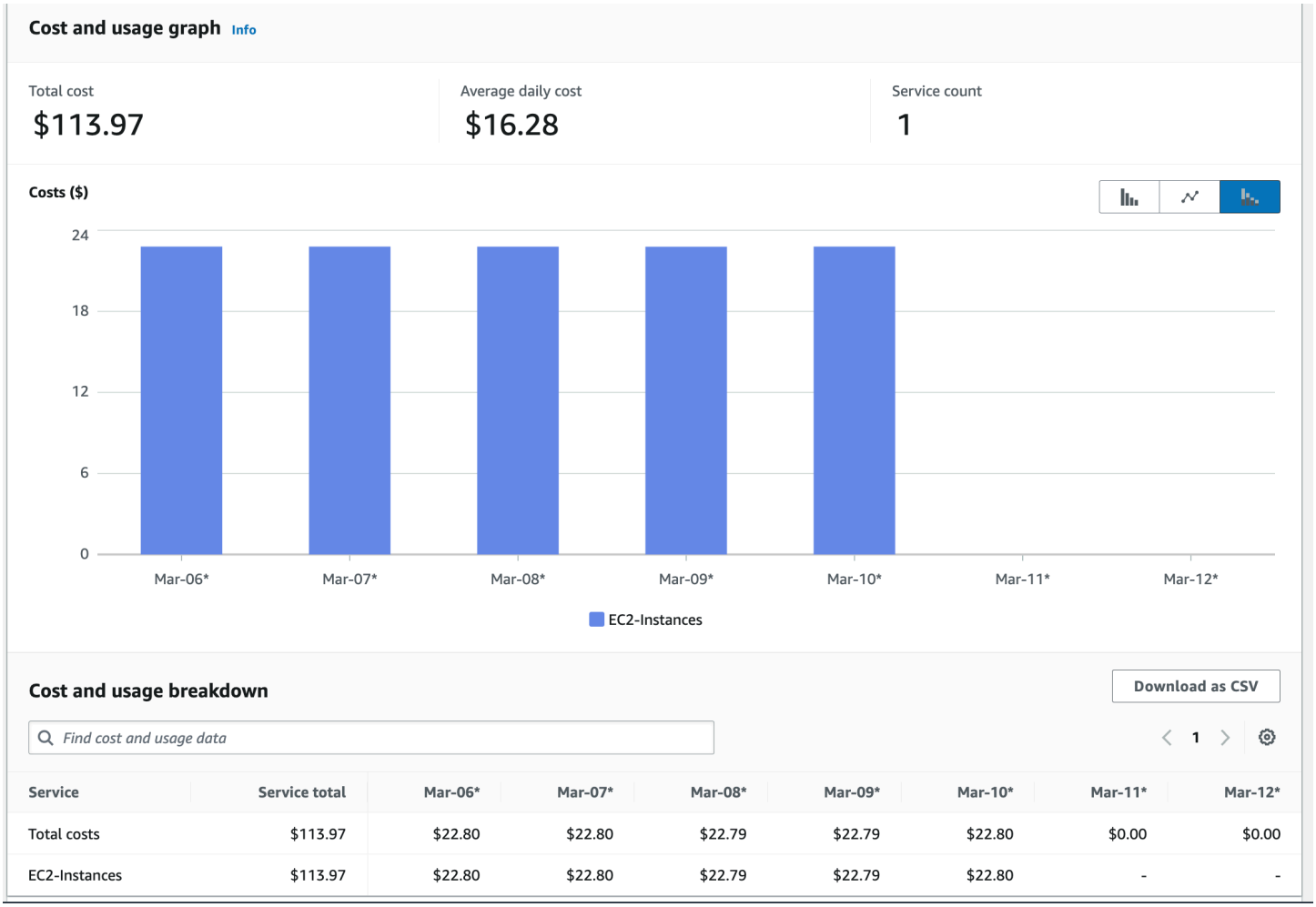


Service	Service total	Mar-06*	Mar-07*	Mar-08*	Mar-09*	Mar-10*	Mar-11*	Mar-12*
Total costs	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36
EC2-Instances	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36

ワークロード A の費用


Cost and usage graph Info
Total cost
$191.50
Average daily cost
$27.36
Service count
1
Costs ($)
28
21
14
7
0
Mar-06*
Mar-07*
Mar-08*
Mar-09*
Mar-10*
Mar-11*
Mar-12*
EC2-Instances
Cost and usage breakdown
Download as CSV
Find cost and usage data
1
Service
Service total
Mar-06*
Mar-07*
Mar-08*
Mar-09*
Mar-10*
Mar-11*
Mar-12*
Total costs
$191.50
$27.36
$27.36
$27.36
$27.36
$27.36
$27.36
$27.36
EC2-Instances
$191.50
$27.36
$27.36
$27.36
$27.36
$27.36
$27.36
$27.36

ワークロード B の費用



その他のリソース

- AWS インスタンスの開始と停止を自動化する (AWS ドキュメントの Instance Scheduler)
- 「Back to Basics: Using an Instance Scheduler to Control Amazon EC2 and Amazon RDS Resource Costs」(YouTube)
- AWS リソースのタグ付け (Tagging AWS Resources User Guide)
- を使用したコストの分析 AWS Cost Explorer (AWS Billing and Cost Management ドキュメント)

Windows ワークロードを適切なサイズに設定する

概要:

適切なサイジングは、最も強力なコスト削減ツールの 1 つです。 は、AWS 最適化とライセンス評価 (AWS OLA) を使用して潜在的なワークロードを確認することから、 を使用して既存のワークロー

ドを確認することまで、適切なサイジング情報を収集するためのさまざまな方法 AWS を提供しますAWS Cost Explorer。

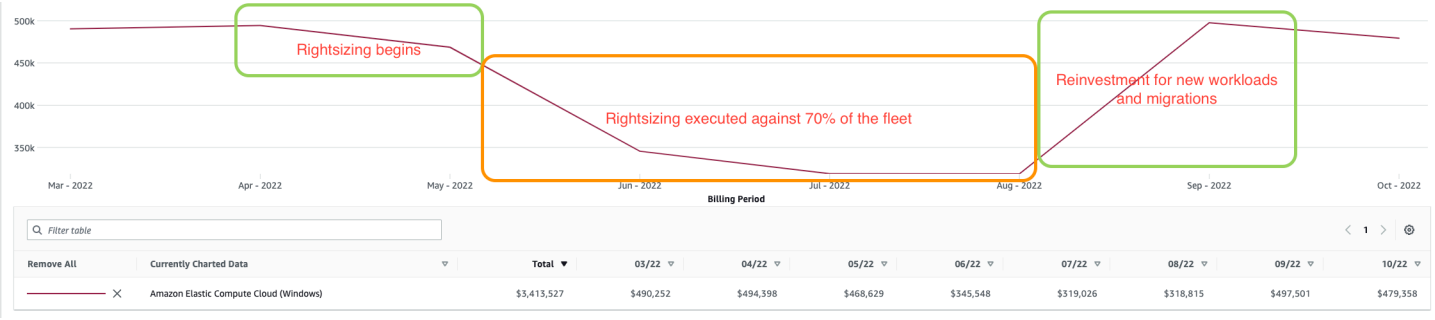
このセクションでは、AWS Compute Optimizer を使用して Amazon EC2 の適切なサイズ設定の機会を特定する方法について説明します。Compute Optimizer は、次のタイプの AWS リソースの過剰プロビジョニングと過少プロビジョニングを防ぐのに役立ちます。

- Amazon Elastic Compute Cloud (Amazon EC2) インスタンスタイプ
- Amazon Elastic Block Store (Amazon EBS) ボリューム
- での Amazon Elastic Container Service (Amazon ECS) サービス AWS Fargate
- Amazon CloudWatch が提供する使用率データに基づく AWS Lambda 関数

コスト最適化シナリオ

適切なサイズ設定の有効性を測定するのは難しい場合があります。適切なサイズ設定の取り組みは、特定のアプリケーション、チーム、または組織全体に向けられる可能性があるためです。たとえば、数千のインスタンスを に移行し AWS、フリートの 90% が Windows ワークロードで構成される組織を考えてみましょう。組織は Compute Optimizer を使用してフリートを分析し、アカウントおよび AWS リージョン全体で大幅な過剰プロビジョニングを検出できます。その後、AWS Systems Manager Automation を使用して、複数のメンテナンスウィンドウを通じてフリートのサイズを適切に設定できます。結果として、組織は、フリートの 70% に対して適切なサイズのインスタンスタイプを調整することができ、35% のコスト削減を実現します。

次のダッシュボードは、このサンプル組織が Compute Optimizer の適切なサイズ設定推奨事項を戦略的に実装したことにより、数か月にわたって達成された削減額を示しています。その目的は、契約終了が近づいているコロケーションデータセンターからの停止した移行を再開するために、既存のワークロードを可能な限り効率的に運用することでした。



コスト最適化の推奨事項

Compute Optimizer を使用してコストを最適化するには、次のステップを実行することをお勧めします。

- Compute Optimizer を有効にする
- Windows ノードのメモリメトリクス収集を有効にする
- Compute Optimizer 推奨事項を使用する
- 適切なサイズ設定のためにインスタンスにタグを付ける
- AWS 請求ツールを操作するためにコスト配分タグを有効にする
- Automation を使用して適切なサイジングレコメンデーションを実装 AWS Systems Manager する
- 代替のサイズ変更方法を検討する
- Cost Explorer で前と後のコストを確認する

Compute Optimizer を有効にする

Compute Optimizer は、組織レベルまたは AWS Organizationsの単一アカウントレベルで有効にできます。組織全体の設定では、すべてのメンバーアカウントのフリート全体の新規および既存のインスタンスの継続的なレポートが提供されます。これにより、適切なサイズ設定を、ポイントインタイムアクティビティではなく、繰り返し行うアクティビティにすることができます。

組織レベル

ほとんどの組織にとって、Compute Optimizer を使用する最も効率的な方法は組織レベルです。これにより、マルチアカウントとマルチリージョンの組織への可視性が提供され、レビューのためにデータを 1 つのソースに一元化できます。組織レベルでこれを有効にするには、以下を実行します。

1. 必要なアクセス許可を持つロールを使用して組織管理アカウントにサインインし、この組織内のすべてのアカウントにオプトインすることを選択します。組織で、すべての機能が有効になっている必要があります。
2. 管理アカウントを有効にしたら、アカウントにサインインし、他のすべてのメンバーアカウントを表示して、その推奨事項を参照できます。

> **Note**
>
> Compute Optimizer の委任管理者アカウントを設定するのがベストプラクティスです。これにより、最小特権の原則を実践できます。そうすれば、組織全体のサービスへのアクセスを提供しながら、組織の管理アカウントへのアクセスを最小限に抑えることができます。

単一アカウントレベル

コストの高いアカウントをターゲットにしているが、 AWS Organizationsにアクセスできない場合は、そのアカウントとリージョンに対して Compute Optimizer を有効にできます。オプトインプロセスの詳細については、Compute Optimizer ドキュメントの「Getting started with AWS Compute Optimizer」を参照してください。

Windows ノードのメモリメトリクス収集を有効にする

メモリメトリクスは、組織内で十分な情報に基づいた適切なサイズ設定の推奨事項を示すために必要な必須メトリクスを Compute Optimizer に提供します。これは、推奨事項を提供する前に CPU、メモリ、ネットワーク、ストレージの分析が行われているためです。

Windows EC2 インスタンスから Compute Optimizer にメモリメトリクスを渡すには、CloudWatch エージェントを有効にし、60 秒ごとにメモリメトリクスを収集するように設定する必要があります。CloudWatch でメモリメトリクスを使用する場合、追加料金はかかりません。

CloudWatch エージェントを有効にしてメモリメトリクスを設定する

ComputeOptimize.yml ファイルをダウンロードします。このファイルを使用して、アカウント内のすべてのインスタンスのメモリ収集を有効にできます。テンプレートファイルは、次のコンポーネントを生成します。

- AWS Systems Manager パラメータストア – メモリメトリクスの収集に必要な CloudWatch エージェントの設定を保存します。
- AWS Identity and Access Management AWS の 管理ポリシー AWS Systems Managerがアタッチされた (IAM) ロール – これは Systems Manager Automation ドキュメント用です。
- AWS Systems Manager ドキュメント – CloudWatch エージェントをインストールして設定します (既存の CloudWatch 設定を置き換えます)。
- AWS Systems Manager ステートマネージャーの関連付け – これにより、Systems Manager ドキュメントをアカウント内のすべてのインスタンスで実行できます。

> **Important**
>
> このテンプレートを実行すると、インスタンスの既存の CloudWatch 設定が上書きされます。

次に、以下の手順を実行します。

1. にサインイン AWS マネジメントコンソール し、CloudFormation コンソールを開きます。
2. ナビゲーションペインで、[Stacks] を選択します。
3. [スタックの作成] を選択し、[既存のリソースを使用 (リソースのインポート)] を選択します。
4. [次へ] を選択します。
5. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。
6. [ファイル] を選択して `ComputeOptimize.yml` ファイルをアップロードします。
7. [次へ] を選択します。
8. [スタックの詳細を指定] ページの [スタック名] に、スタックの名前を入力し、[次へ] を選択します。
9. [リソースを識別] ページで、インポートするリソースの識別子の値を入力します。
10. [リソースをインポート] を選択します。
11. スタックがデプロイされたら、[出力] タブを選択して、関連付けのキー、値、説明を見つけます。

関連付けの進行状況をモニタリングする

1. CloudFormation スタックのデプロイが完了したら、Systems Manager コンソールを開きます。
2. ナビゲーションペインで、[ノード管理] セクションの [ステートマネージャー] を選択します。
3. [関連付け] ページで、関連付けの関連付け ID を選択します。
4. [Execution history (実行履歴の表示)] タブを選択します。
5. [実行 ID] 列で、関連付けの実行 ID を選択します。ステータスは [成功] である必要があります。

CloudWatch でメトリクスを表示する

メトリクスが CloudWatch に入力されるまで、少なくとも 5 分間待つことをお勧めします。

1. CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[メトリクス] セクションを展開し、[すべてのメトリクス] を選択します。
3. メトリクスが CWAgent 名前空間の下に表示されることを確認します。

> **Note**
>
> 新しいインスタンスに設定を適用するには、関連付けを再実行します。

Compute Optimizer 推奨事項を使用する

1 つのアカウントと 1 つのリージョン内で適切なサイズ変更を行うことに焦点を当てた例について考えてみましょう。この例では、Compute Optimizer はすべてのアカウントで組織レベルで有効になっています。適切なサイズ設定は破壊的なプロセスであり、ほとんどの場合、数週間にわたるスケジュールされたメンテナンス期間中にアプリケーション所有者が正確に実行するプロセスであるという点に注意してください。

組織の管理アカウント内から Compute Optimizer に移動する場合は (次の手順を参照)、調査するアカウントを選択できます。この例では、`us-east-1` リージョンの 1 つのアカウントで実行されているインスタンスが 6 つあります。6 つのインスタンスすべてが過剰にプロビジョニングされています。目標は、Compute Optimizer からの推奨事項に基づいてインスタンスのサイズを変更することです。

過剰にプロビジョニングされたインスタンスを特定し、推奨事項の詳細をエクスポートする

1. にサインイン AWS マネジメントコンソール し、Compute Optimizer コンソールを開きます。
2. ナビゲーションペインで、ダッシュボードを選択してください。
3. [ダッシュボード] ページの検索ボックスに、Region=US East (N. Virginia) と入力します。次に、Findings=Over-provisioned と入力します。これらのフィルタを使用すると、`us-east-1` リージョン内の過剰にプロビジョニングされたインスタンスをすべて表示できます。
4. 過剰にプロビジョニングされた EC2 インスタンスの詳細な推奨事項を確認するには、EC2 インスタンスカードまで下にスクロールし、[推奨事項の表示] を選択します。
5. [エクスポート] を選択し、後で使用するためにファイルを保存します。
6. [S3 バケット] には、エクスポートファイルの宛先にする Amazon S3 バケットの名前を入力します。

> **Note**
>
> 今後のレビューのために推奨事項を保存するには、Compute Optimizer が各リージョンで書き込める S3 バケットが必要です。詳細については、Compute Optimizer ドキュメントの「Amazon S3 bucket policy for AWS Compute Optimizer」を参照してください。

7. [フィルタをエクスポート] セクションで、[組織内のすべてのメンバーアカウントの推奨事項を含める] チェックボックスを選択します。
8. [リソースタイプ] で、[EC2 インスタンス] を選択します。
9. [含める列] セクションで、[すべて選択] チェックボックスを選択します。
10. [エクスポート] を選択します。

推奨事項に基づいてインスタンスを選択する

インスタンスの推奨事項は、Compute Optimizer によって収集および分析されたパフォーマンスメトリクスに基づいています。最適なインスタンスを選択するには、インスタンスで実行されているワークロードに注意することが重要です。この例では、最新世代の Amazon EC2 R6i、R5、および T3 インスタンスから選択できることを前提としています。T3 インスタンスはバースト可能で、ネットワーク帯域幅機能は低くなります。R5 インスタンスと R6 インスタンスは、1 時間あたりのコストが同じであり、ほぼ同一です。ただし、R6 インスタンスは、ネットワーク帯域幅容量が大きく、最新世代の Intel プロセッサを搭載しており、R5 と同じコンピューティングフットプリントを提供します。この例では、R6 がサイズ変更に最適なオプションです。

1. Compute Optimizer コンソールで、ナビゲーションバーから [EC2 インスタンスの推奨事項] を選択します。このページでは、現在のインスタンスタイプと、置き換える推奨オプションの比較を示します。
2. 適切なサイズを設定するインスタンスの ID を取得するには、 AWS Organizationsの管理アカウントから Amazon S3 コンソールを開きます。
3. ナビゲーションペインで [バケット] を選択し、エクスポートした結果の保存に使用するバケットを選択します。
4. [オブジェクト] タブで、オブジェクトリストからエクスポートファイルを選択し、[ダウンロード] を選択します。
5. ファイルからインスタンス情報を抽出するには、Microsoft Excel の [データ] タブの [テキストから列へ] ボタンを使用できます。

> **Note**
>
> インスタンス ID は Amazon リソースネーム (ARN) として表されます。必ず区切り文字を「/」に設定して、インスタンス ID を抽出してください。または、スクリプトを記述するか、統合開発環境 (IDE) を使用して ARN をトリミングすることもできます。

6. Excel で、OVER_PROVISIONED インスタンスのみを表示するように検出結果列をフィルタリングします。これらが、適切なサイズ設定の対象となるインスタンスです。
7. 後で簡単にアクセスできるように、テキストエディタでインスタンス ID を保存します。

適切なサイズ設定のためにインスタンスにタグを付ける

ワークロードにタグを付けることは、 AWSでリソースを整理するための強力なツールです。タグを使用すると、コストをきめ細かく可視化し、チャージバックを容易にすることができます。 AWS リソースにタグを追加するための戦略と方法の詳細については、「 リソースのタグ付けに関する AWS ホワイトペーパーのベストプラクティス」を参照してください。 AWSこの例では、AWS タグエディタを使用して、メンテナンス期間中にサイズ変更の対象となる過剰にプロビジョニングされたインスタンス間でタグ付けを調整することができます。このタグを使用して、変更前後のコストを表示することもできます。

1. にサインイン AWS マネジメントコンソール し、サイズ変更の対象となるインスタンスを含むアカウントのAWS Resource Groups コンソールを開きます。
2. ナビゲーションバーの [タグ付け] セクションで、[タグエディタ] を選択します。
3. [リージョン] で、ターゲットリージョンを選択します。
4. [リソースタイプ] で、[AWS::EC2::Instance] を選択します。
5. [リソースを検索] を選択します。
6. [リソースの検索結果] ページで、適切なサイズに設定するインスタンスをすべて選択し、[選択したリソースのタグを管理する] を選択します。
7. [タグを追加] を選択します。
8. [タグキー] には、Rightsizing と入力します。[タグ値] には、enabled と入力します。そして、[タグの変更を確認して適用する] を選択します。

> **Note**
>
> チームやビジネスユニットなどの追加のメタデータを含めれば、後で Cost Explorer でフィルタリングできます。

ユーザー定義のタグを作成してリソースに適用した後、アクティブ化のためにタグがコスト配分タグページに表示されるまでに最大で 24 時間かかる場合があります。アクティブ化のためにタグを選択した後、タグが有効になるまでにさらに 24 時間かかることがあります。

上級ユーザーは、ターゲットアカウントとリージョン内で AWS CloudShell を使用して、複数のインスタンスにタグを付けることができます。例えば、次のようになります。

```
bash
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="type-m5"
# Get a list of instance IDs
INSTANCE_IDS=$(aws ec2 describe-instances –query
 "Reservations[].Instances[].InstanceId" –output text)
# Loop through each instance ID and add the tag
for INSTANCE_ID in $INSTANCE_IDS; do
 aws ec2 create-tags –resources $INSTANCE_ID –tags Key=$TAG_KEY,Value=$TAG_VALUE
done
```

AWS 請求ツールを操作するためにコスト配分タグを有効にする

ユーザー定義のコスト配分タグをアクティブ化することをお勧めします。これにより、 AWS 請求ツール (Cost Explorer や など) で Rightsizing タグが認識され、フィルタリングできるようになります AWS Cost and Usage Report。これを有効にしない場合、タグフィルタリングオプションとデータは使用できなくなります。コスト配分タグの使用の詳細については、 AWS Billing and Cost Management ドキュメントの「ユーザー定義のコスト配分タグのアクティブ化」を参照してください。

1. にサインイン AWS マネジメントコンソール し、 AWS Billing コンソールを開きます。
2. ナビゲーションペインの [請求] セクションで、[コスト配分タグ] を選択します。
3. [ユーザー定義のコスト配分タグ] タブで、Rightsizing と入力します。

4. Rightsizing タグキーを選択し、[アクティブ化] を選択します。

24 時間後、Cost Explorer にタグが表示されます。

Systems Manager Automation を使用して適切なサイズ設定の推奨事項を実装する

サイズ変更は、インスタンスを停止して開始する必要があるシナリオです。このシナリオでは、メンテナンス期間にこの中断を処理し、独自のサイズ変更を処理するために別のチームが必要になる場合があります。インスタンスタイプを変更する前に、Amazon EC2 ドキュメントの「Considerations for compatible instance types」を確認してください。

このセクションのステップ例では、AWS-ResizeInstance という Systems Manager Automation ドキュメントを使用して、アカウントおよびリージョンごとに適切なサイズ設定の推奨事項を実装します。ほとんどの組織はさまざまな目的で異なるインスタンスタイプを必要とするため、このアプローチはほとんどの組織にとって一般的です。同じ `AWS-ResizeInstance` オートメーションドキュメントを使用して、単一アカウントおよびマルチアカウントのデプロイを対象にすることもできます。

1. にサインイン AWS マネジメントコンソール し、Systems Manager コンソールを開きます。
2. ナビゲーションペインの [共有リソース] セクションで、[ドキュメント] を選択します。
3. 検索バーに AWS-ResizeInstance と入力し、検索結果から AWS-ResizeInstance を選択します。
4. [Execute automation (自動化の実行)] を選択してください。
5. [オートメーションランブックを実行する] ページで、[シンプルな実行] を選択します。
6. [入力パラメータ] セクションで、InstanceId および InstanceType と入力します。残りはデフォルト値のままにしておきます。
7. [実行] を選択し、オートメーションがインスタンスタイプを変更するステップを実行するのを待ちます。

代替のサイズ変更方法を検討する

起動テンプレートを使用してインスタンスをデプロイする場合は、起動テンプレートを適切なサイズのインスタンスタイプで更新し、インスタンスの更新を実行してインスタンスを適切なサイズのバージョンに置き換えることができます。

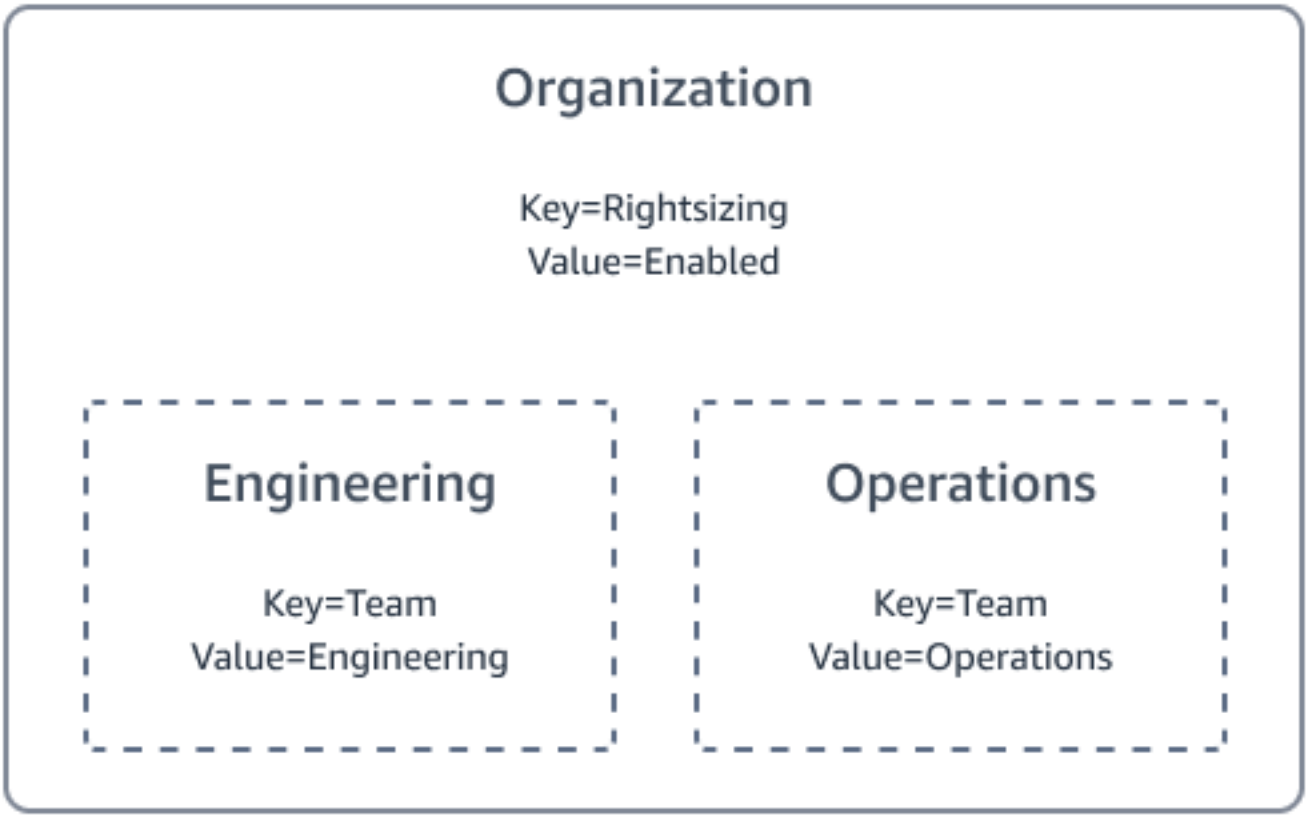
複数のアカウントとリージョンで適切なサイズ設定プロセスを使用する場合は、カスタム Systems Manager Automation ドキュメントを作成する必要があります。このドキュメントでは、複数のインスタンスをパラメータとしてフィードし、同じ宛先インスタンスタイプに移行するインスタンスを対

象にすることができます (例えば、ソースインスタンスタイプに関係なく、t3a.medium に移行するすべてのインスタンス)。

Cost Explorer で前と後のコストを確認する

リソースのサイズを適切に設定したら、Cost Explorer を使用し、[ライトサイジング] タグを使用して、前と後のコストを表示できます。リソースタグを使用してコストを追跡できることを思い出してください。複数のタグレイヤーを使用することで、コストをきめ細かく可視化できます。このガイドで説明する例では、[ライトサイジング] タグを使用して、すべてのターゲットインスタンスに汎用タグを適用します。次に、[チーム] タグを使用して、リソースをさらに整理します。次のステップでは、アプリケーションタグを導入して、特定のアプリケーションを運用する際のコストへの影響をさらに示します。

次の図は、組織のタグ構造を示しています。



運用チームが所有する本番用ウェブサーバーのサイズを適切に設定するビジネスの例を考えてみましょう。Cost Explorer では、Rightsizing タグは enabled に設定され、Team タグは operations に設定されています。この例では、適切なサイズ設定作業により、運用コストが 1 時間あたり 0.89 セントから 0.28 セントに削減されます。1 か月あたり 744 時間と仮定すると、適切なサイズ設定の前の年間コストは 7,945.92 USD です。適切なサイズ設定の後、年間コストは 2,499.84 USD に削減され

ます。これにより、年間ワークロードコストが 68.5% 削減されることになります。これが大規模な組織全体に及ぼす影響を想像してみてください。これはサンプル環境で行われ、インスタンスはほとんどアイドル状態であることに注意してください。本番環境では、10～35% の節約が見込まれます。

次に、エンジニアリングチームが所有する本番用踏み台ホストの適切なサイズ設定の影響について検討します。Cost Explorer では、Rightsizing タグは enabled に設定され、Team タグは engineering に設定されています。この例では、適切なサイズ設定作業により、コストが 1 時間あたり 0.75 セントから 0.44 セントに削減されます。1 か月あたり 744 時間と仮定すると、適切なサイズ設定の前の年間コストは 6,696.00 USD です。適切なサイズ設定の後、年間コストは 3,928.32 USD に削減されます。

複数のタグを使用する場合は、データフィルタリングしてコストの詳細まで確認することができます。この例では、Team タグによってノイズが削減され、チームレベルで影響を確認できるようになります。Rightsizing タグが有効になっているため、そのタグの値が enabled になっているインスタンス、または値が存在しないインスタンスをフィルタリングすることもできます。これにより、特に Cost Explorer レベルで管理アカウント (支払者) に表示される場合、適切なサイズ設定作業のグローバルビューを提供できます。このビューでは、すべてのアカウントとインスタンスを表示できます。

Rightsizing タグが enabled に設定されている単一アカウントレベルでの例について考えてみましょう。 運用コストは 1 時間あたり 1.64 USD から 0.72 USD に減少します。1 か月あたり 744 時間と仮定すると、適切なサイズ設定の前の年間コストは 14,641.92 USD です。適切なサイズ設定の後、年間コストは 6,428.16 USD に削減されます。これにより、このアカウントのコンピューティングコストが 56% 削減されます。

適切なサイズ設定ジャーニーを開始する前に、次の点を考慮してください。

- AWS には、コスト削減のための多くのオプションがあります。これには、 が移行前にオンプレミスインスタンス AWS を確認する AWS OLA が含まれます AWS。OLA AWS は、適切なサイジングの推奨事項とライセンスガイダンスも提供します。
- Savings Plans を購入する前に、適切なサイズ設定をすべて完了してください。これにより、Savings Plans コミットメントでの過剰購入を回避できます。

推奨事項

次のステップとして以下の手順の実行を推奨します。

1. 既存のランドスケープを確認し、Amazon EBS gp2 ボリュームを gp3 ボリュームに変換することを検討します。

2. Savings Plans を確認します。

その他のリソース

- AWS Compute Optimizer (AWS ドキュメント)
- AWS リソースのタグ付けのベストプラクティス (AWS ホワイトペーパー)
- AWS Compute OptimizerAWS Trusted Advisor との間でデータを収集する方法 AWS Organizations (YouTube)
- 「Optimizing performance and reducing licensing costs: Leveraging AWS Compute Optimizer for Amazon EC2 SQL Server instances」(Microsoft Workloads on AWS ブログ)

Windows ワークロードに適したインスタンスタイプを選択する

概要:

クラウドで運用されているワークロードとオンプレミス環境との大きな違いは、過剰プロビジョニングの実践です。オンプレミス用の物理ハードウェアを購入する場合は、事前に決められた期間 (通常は 3~5 年) にわたって継続することが予想される設備投資を行います。ハードウェアの耐用年数中に予想される増加に対応するために、ハードウェアはワークロードが現在必要とするよりも多くのリソースで取得されます。したがって、物理ハードウェアは多くの場合、実際のワークロードのニーズをはるかに超えて過剰にプロビジョニングされます。

仮想マシン (VM) テクノロジーは、余剰ハードウェアリソースを利用する効果的な手段として登場しました。管理者は vCPU と RAM で VM を過剰にプロビジョニングし、各 VM に未使用のリソースを割り当てることで、ハイパーバイザーがビジーサーバーとアイドルサーバー間の物理リソース使用状況を管理できるようにしました。VM を管理する場合、各 VM に割り当てられた vCPU リソースと RAM リソースは、実際の使用状況の指標ではなく、リソースガバナーとして機能していました。VM リソースの過剰割り当ては、使用可能なコンピューティングリソースの 3 倍を簡単に超える可能性があります。

Amazon Elastic Compute Cloud (Amazon EC2) は、基盤となるハードウェアに VM を過剰にプロビジョニングしないようにします。それは不要だからです。クラウドコンピューティングは、資本コストではなく運用コストであり、使用した分に対してのみ支払います。将来的にワークロードにより多くのリソースが必要になる場合は、事前に準備するのではなく、実際に必要になったときにプロビジョニングします。

適切な Amazon EC2 インスタンスタイプを選択するためのオプションは数百があります。Windows ワークロードをクラウドに移行する予定の場合、 は AWS OLA AWS を提供し、現在のワークロードをよりよく理解し、 でのパフォーマンスの例を提供します AWS。 AWS OLA 分析は、適切な EC2 インスタンスタイプとサイズを実際のオンプレミス使用量と一致させることを目的としています。

Amazon EC2 で実行されているワークロードが既にあり、コスト最適化戦略を検討している場合、本ガイドのこのセクションは、Amazon EC2 インスタンス間の違いと、一般的な Windows ワークロードへの適用性を特定するのに役立ちます。

コスト最適化の推奨事項

EC2 インスタンスタイプのコストを最適化するには、以下を実行することをお勧めします。

- ワークロードに適したインスタンスファミリーを選択する
- プロセッサアーキテクチャ間の料金の違いを理解する
- EC2 世代間の価格性能比の違いを理解する
- 新しいインスタンスに移行する
- バースト可能インスタンスを使用する

ワークロードに適したインスタンスファミリーを選択する

ワークロードに適したインスタンスファミリーを選択することが重要です。

Amazon EC2 インスタンスは、次のさまざまなグループに分かれています。

- 汎用
- コンピューティングの最適化
- メモリ最適化
- 高速コンピューティング
- ストレージの最適化
- HPC 最適化

ほとんどの Windows ワークロードは、次のカテゴリに該当します。

- 汎用
- コンピューティングの最適化

- メモリ最適化

これをさらに単純化するには、各カテゴリのベースライン EC2 インスタンスを検討します。

- コンピューティングの最適化 – C6i
- 凡用 – M6i
- メモリ最適化 – R6i

前世代の EC2 インスタンスでは、プロセッサタイプにわずかな違いがありました。例えば、C5 コンピューティング最適化インスタンスは、M5 汎用インスタンスまたは R5 メモリ最適化インスタンスよりも高速なプロセッサを備えています。最新世代の EC2 インスタンス (C6i、M6i、R6i、C6a、M6a、R6a) はすべて、インスタンスファミリー間で同じプロセッサを使用します。プロセッサは最新世代のインスタンス間で一貫しているため、インスタンスファミリー間の料金差は、RAM の量に大きく左右されるようになりました。インスタンスの RAM が多いほど、コストが高くなります。

次の例は、`us-east-1` リージョンで実行されている Intel ベースの 4 vCPU インスタンスの時間単位の料金を示しています。

インスタンス	vCPU	RAM	時間料金
c6i.xlarge	4	8	0.17 USD
m6i.xlarge	4	16	0.19 USD
r6i.xlarge	4	32	0.25 USD

> **Note**
>
> 料金は、`us-east-1` リージョンのオンデマンドの時間単位の料金に基づいています。

バースト可能インスタンス

クラウドコンピューティングでは、未使用のコンピューティングリソースをオフにして課金を回避するのがベストプラクティスですが、必要になるたびにすべてのワークロードをオフにしたりオンにし

たりできるわけではありません。一部のワークロードは長期間アイドル状態のままですが、1 日 24 時間アクセス可能である必要があります。

バースト可能インスタンス (T3) は、コンピューティングコストを低く抑えながら、急増する、または使用率の低いワークロードを 1 日中オンラインで維持する方法を提供します。バースト可能 EC2 インスタンスには、インスタンスが短期間使用できる vCPU リソースの最大量があります。これらのインスタンスは、バースト可能 CPU クレジットに基づくシステムを使用します。これらのクレジットは、1 日を通してアイドル期間中に蓄積されます。バースト可能インスタンスは、さまざまな vCPU 対 RAM 比率を提供するため、場合によってはコンピューティング最適化インスタンスの代替となり、また場合によっては他の汎用インスタンスの代替となります。

次の例は、us-east-1 リージョンで実行されている T3 インスタンス (つまりバースト可能インスタンス) の時間単位の料金を示しています。

インスタンス	vCPU	RAM (GB)	時間料金
t3.nano	2	0.5	0.0052 USD
t3.micro	2	1	0.0104 USD
t3.small	2	2	0.0208 USD
t3.medium	2	4	0.0416 USD
t3.large	2	8	0.0832 USD
t3.xlarge	4	16	0.1664 USD
t3.2xlarge	8	32	0.3328 USD

> **Note**
>
> 料金は、us-east-1 リージョンのオンデマンドの時間単位の料金に基づいています。

プロセッサアーキテクチャ間の料金の違いを理解する

Intel プロセッサは、EC2 インスタンスの登場以来、その標準となっています。C5、M5、R5 などの旧世代の EC2 インスタンスでは、Intel をプロセッサアーキテクチャとして明示していません (デ

フォルトだったため)。C6i、M6i、R6i などの新世代の EC2 インスタンスには、Intel プロセッサの使用を示す「i」が含まれています。

プロセッサアーキテクチャの注釈の変更は、追加のプロセッサオプションの導入によるものです。Intel に最も近いプロセッサは AMD です (「a」で示されます)。AMD EPYC プロセッサは、同じ x86 アーキテクチャを使用し、Intel プロセッサと同様のパフォーマンスを低コストで提供します。次の料金例に示すように、AMD EC2 インスタンスでは、Intel インスタンスと比較してコンピューティングコストが約 10% 削減されます。

Intel インスタンス	時間料金	AMD インスタンス	料金	差 (%)
c6i.xlarge	0.17 USD	c6a.xlarge	0.153 USD	10%
m6i.xlarge	0.192 USD	m6a.xlarge	0.1728 USD	10%
r6i.xlarge	0.252 USD	r6a.xlarge	0.2268 USD	10%

> **Note**
>
> 料金は、`us-east-1` リージョンのオンデマンドの時間単位の料金に基づいています。

3 番目の主要なプロセッサアーキテクチャオプションは、EC2 インスタンス上の AWS Graviton プロセッサです (「g」で示されます)。によって設計された Graviton プロセッサは AWS、Amazon EC2 で最高の価格パフォーマンスを提供します。現在の Graviton プロセッサは、Intel プロセッサよりも 20% 安いだけでなく、パフォーマンスも 20% 以上向上しています。次世代の Graviton プロセッサは、このパフォーマンスの差をさらに拡大することが予想され、テストではパフォーマンスがさらに 25% 向上することがわかっています。

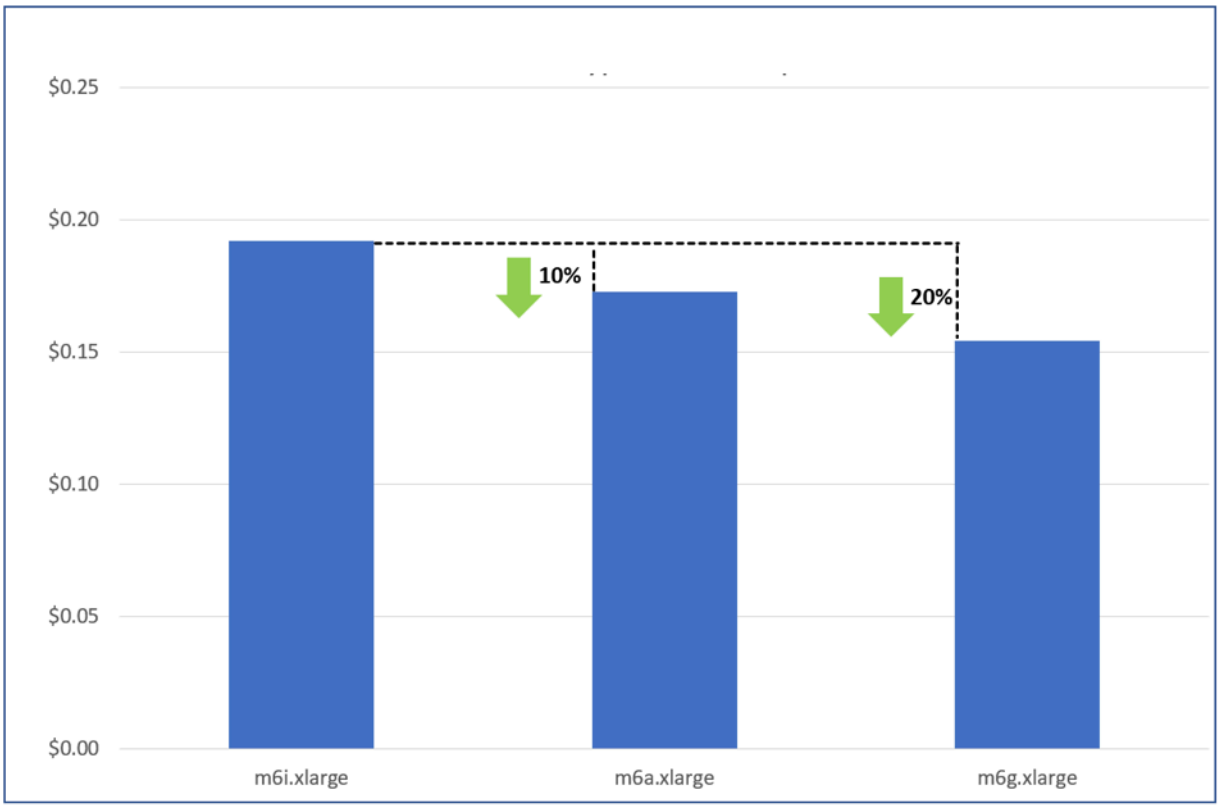
Windows Server は、ARM アーキテクチャに基づく Graviton プロセッサでは実行できません。実際、Windows Server は x86 プロセッサでのみ動作します。Windows Server に Graviton ベースのインスタンスを使用して価格性能比を 40% 向上させることはできませんが、特定の Microsoft ワークロードで Graviton プロセッサを使用することはできます。例えば、新しいバージョンの .NET は Linux で実行できます。つまり、これらのワークロードは ARM プロセッサを使用し、より高速で手頃な価格の Graviton EC2 インスタンスからメリットを得ることができます。

次の例は、us-east-1 リージョンで実行されている Graviton インスタンスの時間単位の料金を示しています。

Intel インスタンス	時間料金	Graviton インスタンス	時間料金	差 (%)
c6i.xlarge	0.17 USD	c6g.xlarge	0.136 USD	20%
m6i.xlarge	0.192 USD	m6g.xlarge	0.154 USD	20%
r6i.xlarge	0.252 USD	r6g.xlarge	0.2016 USD	20%

> **Note**
>
> 料金は、us-east-1 リージョンのオンデマンドの時間単位の料金に基づいています。

次のグラフは、M シリーズインスタンスの料金を比較したものです。



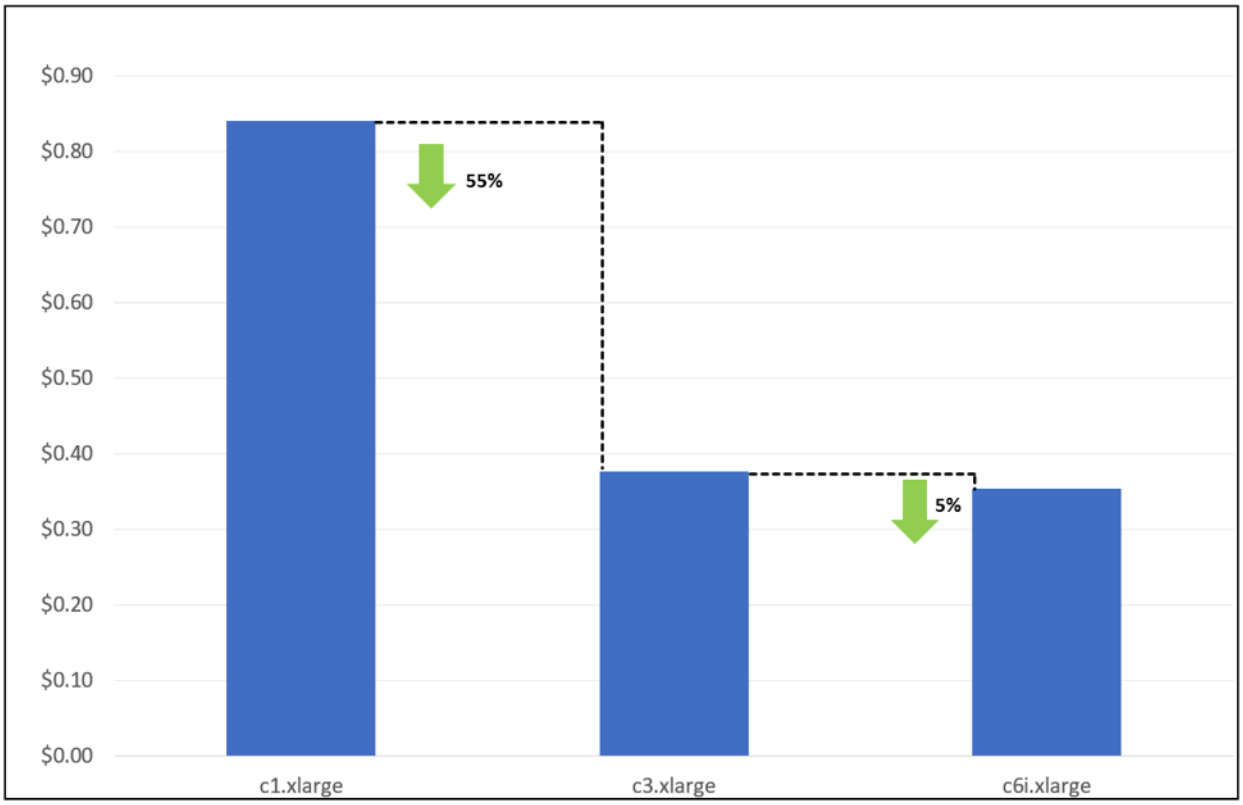
EC2 世代間の価格性能比の違いを理解する

Amazon EC2 の最も一貫した特徴の 1 つは、各新世代が前世代よりも優れた価格性能比を提供することです。次の表に示すように、新世代の EC2 インスタンスの料金は、後続のリリースごとに下がります。

コンピューティング最適化インスタンス	時間料金	汎用インスタンス	時間料金	メモリ最適化インスタンス	時間料金
C1.xlarge	0.52 USD	M1.xlarge	0.35 USD	r1.xlarge	該当なし
C3.xlarge	0.21 USD	M3.xlarge	0.266 USD	r3.xlarge	0.333 USD
C5.xlarge	0.17 USD	M5.xlarge	0.192 USD	r5.xlarge	0.252 USD

> **Note**
>
> 料金は、`us-east-1` リージョンのオンデマンドの時間単位の料金に基づいています。

次のグラフは、さまざまな世代の C シリーズインスタンスのコストを比較したものです。



ただし、次の表に示すように、第 6 世代のインスタンスは第 5 世代と同じ料金です。

コンピューティング最適化インスタンス	時間料金	汎用インスタンス	時間料金	メモリ最適化インスタンス	時間料金
C5.xlarge	0.17 USD	M5.xlarge	0.192 USD	r5.xlarge	0.252 USD
C6i.xlarge	0.17 USD	M6i.xlarge	0.192 USD	r6i.xlarge	0.252 USD

> **Note**
>
> 料金は、`us-east-1` リージョンのオンデマンドの時間単位の料金に基づいています。

コストは同じですが、プロセッサの高速化、ネットワークスループットの向上、Amazon Elastic Block Store (Amazon EBS) のスループットと IOPS の向上により、新世代では優れた価格性能比が提供されます。

価格性能比の最も重要な改善の 1 つは、X2i インスタンスの強化です。この世代のインスタンスは、前世代よりも最大 55% 高い価格性能比を提供します。次の表に示すように、x2iedn は、パフォーマンスのあらゆる面で改善を示しています (すべて前世代と同じ料金)。

インスタンス	時間料金	vCPU	RAM	プロセッサ速度	インスタンスストレージ	ネットワーク	Amazon EBS スループット	EBS IOPS
x1e.2xlarge	1.66 USD	8	244	2.3 GHz	237 GB SSD	10 Gbps	125 MB/秒	7400
x1iedn.2xlarge	1.66 USD	8	256	3.5 GHz	240 GB NVMe SSD	25 Gbps	2500 MB/秒	65000

> **Note**
>
> 料金は、us-east-1 リージョンのオンデマンドの時間単位の料金に基づいています。

シナリオ例

配送車両を追跡し、SQL Server のパフォーマンスを向上させたいと考えている分析会社の例について考えてみましょう。MACO SME がこの会社のパフォーマンスのボトルネックを確認した後、会社は x1e.2xlarge インスタンスから x2iedn.xlarge インスタンスに移行します。新しいインスタンスサイズは小さくなりますが、x2 インスタンスの機能強化により、バッファプール拡張機能の使用を通じて SQL Server のパフォーマンスと最適化が向上します。これにより、同社は SQL Server Enterprise Edition から SQL Server Standard Edition にダウングレードできます。また、これにより同社は SQL Server ライセンスを 8 vCPU から 4 vCPU に削減することができます。

最適化前:

サーバー	EC2 インスタンス	SQL Server エディション	月額コスト
ProdDB1	x1e.2xlarge	Enterprise	3,918.64 USD

サーバー	EC2 インスタンス	SQL Server エディション	月額コスト
ProdDB2	x1e.2xlarge	Enterprise	3,918.64 USD
合計			7,837.28 USD

最適化後:

サーバー	EC2 インスタンス	SQL Server エディション	月額コスト
ProdDB1	x2iedn.xlarge	標準	1,215.00 USD
ProdDB2	x2iedn.xlarge	標準	1,215.00 USD
合計			2,430.00 USD

すべてを合わせると、x1e.2xlarge インスタンスから x2iedn.xlarge インスタンスへの変更により、このシナリオ例では、本番用データベースサーバーで 1 か月あたり 5,407 USD を節約できます。これにより、ワークロードの合計コストが 69% 削減されます。

> **Note**
>
> 料金は、`us-east-1` リージョンのオンデマンドの時間単位の料金に基づいています。

新しいインスタンスに移行する

旧世代の Amazon EC2 は Xen ハイパーバイザーで動作し、新世代は AWS Nitro System で動作します。Nitro System は、ホストハードウェアのほぼすべてのコンピューティングリソースとメモリリソースをインスタンスに提供します。これにより、全体的なパフォーマンスが向上します。Xen から Nitro ベースのインスタンスに移行する場合は、特別な考慮事項があります。例えば、AWS Windows AMI は、Microsoft インストールメディアで使用されるデフォルト設定とカスタマイズで設定されます。カスタマイズには、最新世代のインスタンスタイプ (Nitro System 上に構築されたインスタンス) をサポートするドライバーと設定が含まれます。

カスタム Windows AMI から、または 2018 年 8 月より前に作成された Amazon 提供の Windows AMI からインスタンスを起動する場合は、Amazon EC2 ドキュメントの「Migrate to latest generation instance types」の手順を完了することをお勧めします。

バースト可能インスタンスを使用する

バースト可能インスタンスはコンピューティングコストを節約する良い方法ですが、以下のシナリオでは避けることをお勧めします。

- デスクトップエクスペリエンスを使用する Windows Server の最小仕様には、2 GB の RAM が必要です。RAM の最小容量を満たしていないため、Windows Server で t3.micro または t3.nano インスタンスを使用することは避けてください。
- ワークロードが急増しているが、バーストクレジットを構築するのに十分な時間アイドル状態にならない場合は、通常の EC2 インスタンスを使用する方が、バースト可能インスタンスを使用するよりも効率的です。CPU クレジットをモニタリングしてこれを確認することをお勧めします。
- ほとんどのシナリオでは、SQL Server でバースト可能インスタンスを使用しないことをお勧めします。SQL Server のライセンスは、インスタンスに割り当てられた vCPU の数に基づいています。SQL Server が 1 日の大半アイドル状態の場合、完全に活用されていない SQL ライセンスに対して料金が発生します。これらのシナリオでは、複数の SQL Server インスタンスをより大きなサーバーに統合することをお勧めします。

次の手順

Amazon EC2 Windows インスタンスのコストを最適化するには、次のステップを実行することをお勧めします。

- 最高の価格性能比を得るには、最新世代の EC2 インスタンスを使用します。
- AMD プロセッサで EC2 インスタンスを使用すると、コンピューティングコストを 10% 削減できます。
- ワークロードに一致する EC2 インスタンスタイプを選択して、リソース使用率を最大化します。

次の表は、Windows ワークロードの一般的な開始点の例を示しています。SQL Server ワークロードを強化するためのインスタンスストレージボリュームや、vCPU と RAM の比率がはるかに大きい EC2 インスタンスなど、追加のオプションを使用できます。ワークロードを徹底的にテストし、AWS Compute Optimizer などのモニタリングツールを使用して必要な調整を行うことをお勧めします。

ワークロード	標準	オプションです。
Active Directory	T3、M6i	R6i
ファイルサーバー	T3、M6i	C6i
ウェブサーバー	T3、C6i	M6i、R6i
SQL Server	R6i	x2iedn、X2iezn

EC2 インスタンスタイプを変更する必要がある場合、プロセスには通常、単純なサーバーの再起動のみが含まれます。詳細については、Amazon EC2 ドキュメントの「Amazon EC2 インスタンスタイプの変更」を参照してください。

インスタンスタイプを変更する前に、次の点を考慮することをお勧めします。

- インスタンスタイプを変更する前に、Amazon EBS–backed インスタンスを停止する必要があります。インスタンスが停止している間のダウンタイムを計画しておいてください。インスタンスを停止し、インスタンスタイプの変更を行うと、数分かかります。インスタンスを再起動すると、アプリケーションの起動スクリプトによってかかる時間が変動する場合があります。詳細については、Amazon EC2 ドキュメントの「Amazon EC2 インスタンスの停止と起動」を参照してください。
- インスタンスを停止して起動すると、 はインスタンスを新しいハードウェア AWS に移動します。インスタンスにパブリック IPv4 アドレスがある場合、 はそのアドレスを AWS 解放し、インスタンスに新しいパブリック IPv4 アドレスを付与します。変更されないパブリック IPv4 アドレスが必要な場合は、Elastic IP アドレスを使用します。
- 休止状態が有効になっているインスタンスのインスタンスタイプを変更することはできません。
- [Spot Instance] (スポットインスタンス) のインスタンスタイプを変更することはできません。
- インスタンスが Auto Scaling グループにある場合、Amazon EC2 Auto Scaling はインスタンスを異常と判断して停止し、場合によってはそれを終了して代わりのインスタンスを起動します。インスタンスタイプを変更するときに、そのグループのスケーリングプロセスを中断することで、これを防ぐことができます。詳細については、Amazon EC2 Auto Scaling ドキュメントの「Amazon EC2 Auto Scaling プロセスの中断と再開」を参照してください。
- NVMe インスタンスストアボリュームを使用してインスタンスのインスタンスタイプを変更すると、Amazon マシンイメージ (AMI) またはインスタンスブロックデバイスマッピングで指定されていない場合でもすべての NVMe インスタンスストアボリュームが利用可能であるため、追加のイ

ンスタンスストアボリュームが更新されたインスタンスに存在するようになる可能性があります。それ以外の場合、更新したインスタンスには、元のインスタンスの起動時に指定したのと同じ数のインスタンスストアボリュームが設定されます。

その他のリソース

- Amazon EC2 インスタンスタイプ (AWS ドキュメント)
- AWS Optimization and Licensing Assessment (AWS ドキュメント)

Windows および SQL Server ワークロードのライセンスを持ち込む

概要:

Microsoft ワークロードと既存のエンタープライズライセンス契約に多額の投資がある場合は、ライセンス込み (提供 AWS) や Bring Your Own License (BYOL) AWS オプションなど、これらのワークロードをサポートするいくつかのオプションから選択できます。Amazon EC2 専有ホストを使用すると、既存の Microsoft ライセンス契約を最大限に活用し、Windows Server を AWSに持ち込むことができます。これにより、Amazon EC2 インスタンスのコストを最大 50% 削減できます。Windows ライセンスはインスタンスコストの約半分を占めるため、Windows Server を Dedicated Hosts AWS で に持ち込むと、大幅なコスト削減につながります。Windows Server はデフォルト (共有) テナンシーに持ち込むことができないため、Windows Server の既存のライセンスを使用する場合は、Dedicated Hosts が最適です AWS。

専有ホストは Windows Server BYOL インスタンス専用ではありません。また、既存の SQL Server ワークロードのオンプレミスライセンスに合わせて柔軟に対応できます。専有ホストは、基盤となるサーバーの物理コアを公開し、物理コアレベルで SQL Server をライセンスできるようにします。これは、SQL Server ライセンスがインスタンスに割り当てられた仮想 CPU の数に基づいているデフォルトの (共有) テナンシーでは不可能です。この機能を使用すると、オンプレミスのライセンス戦略と一致する方法で AWS で SQL Server ワークロードをライセンスできます。したがって、対象となる Windows ライセンスを使用することで、インスタンスコストの削減に加えて、デフォルトの (共有) テナンシーと比較して SQL Server ライセンスコストを最大 50% 削減できます。このシナリオの詳細については、このガイドの「SQL Server ライセンスを理解する」セクションを参照してください。

Amazon EC2 専有ホスト

Amazon EC2 Dedicated Host は、基本的に が EC2 コンピューティングサービスの実行 AWS に使用するのと同じ EC2 ホストです。違いは、これらのホストは完全に単一のお客様専用であり、基盤となる物理インフラストラクチャへの排他的なアクセスを提供するという点です。専有ホストを使用すると、他の AWS のお客様とリソースを共有するのではなく、完全にお客様専用のハードウェアでインスタンスを実行できます。これにより、クラウドリソースをより詳細に制御でき、Windows Server や SQL Server などの独自のソフトウェアライセンスを AWSに持ち込むことでコストを削減できます。

以下に留意してください。

- 専有ホストは、完全に単一のお客様専用の物理サーバーです。専有ホストのソケットと物理コアを可視化することで、ソケット単位、コア単位、VM 単位のソフトウェアライセンス契約などのライセンスコンプライアンス要件に対処できます。
- 同じインスタンスファミリーの複数のインスタンスサイズをサポートできる専有ホストは、異種専有ホストと呼ばれます。これらのインスタンスファミリーには T3、A1、C5、M5、R5、C5n、R5n、M5n が含まれます。対照的に、他のインスタンスファミリーは、同じ専有ホストにおける 1 つのインスタンスサイズのみをサポートしています。これらは同種専有ホストと呼ばれます。
- 専有ホストはホストごとに課金されます。つまり、実行されているインスタンスの数に関係なく、専有ホストごとに課金されます。専有ホストの料金は、インスタンスファミリー、リージョン、選択した支払いオプションによって異なります。ワークロードに最適な設定を選択して、希望するパフォーマンスとコストの成果を達成できます。

この図は、共有テナンシーインスタンスと専有ホストの違いを示しています。

同種専有ホスト

M6i 専有ホストを使用するシナリオを考えてみましょう。M6i および R6i 専有ホストには、2 個のソケット、64 個の物理コアがあり、同じサイズのインスタンスタイプをサポートしています。これらは同種専有ホストと呼ばれます。つまり、単一の M6i 専有ホストで起動できるインスタンスの数は、インスタンスサイズによって異なります。

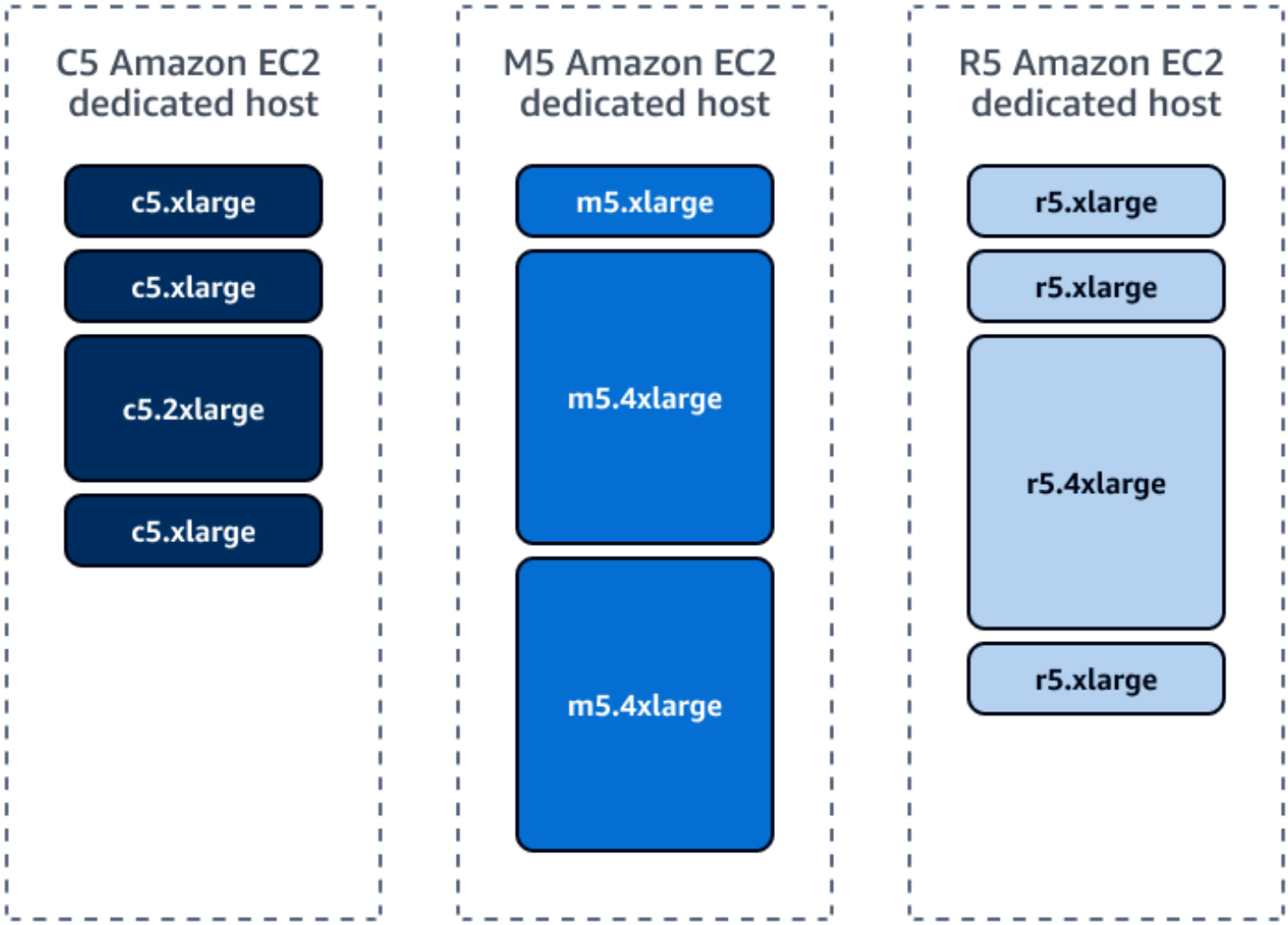
例えば、次のようになります。

- xlarge (4 vCPU) の場合は、この専有ホストで最大 32 個の m6i.xlarge インスタンスを起動できます。
- 8xlarge (32 vCPU) の場合は、この専有ホストで最大 4 個の m6i.8xlarge インスタンスを起動できます。
- metal (128 vCPU) の場合は、この専有ホストで最大 1 個の m6i.metal インスタンスを起動できます。

次の図は、M6 インスタンスの専有ホストオプションを示しています。

異種専有ホスト

同じホストで複数のインスタンスサイズをサポートする専有ホストは、異種 Amazon EC2 専有ホストと呼ばれます。次の図は、2xlarge、xlarge、4xlarge などのさまざまなインスタンスサイズを持つ、C5、M5、および R5 専有ホストの例を示しています。



専有ホストの管理

Amazon EC2 専有ホストの管理については、次の点を考慮することをお勧めします。

- 専有ホストを最大限に活用するために、組織内の複数のアカウント間で単一のホストを共有できます。ホスト共有により、リソースの最適化が可能になり、ホストで使用可能なすべてのスロットを使用することでコストを削減できます。専有ホストをビジネスユニット間で共有することで、ワークロード間の分離を維持しながら、IT インフラストラクチャを一元化し、リソース使用率を向上させることができます。の組織に属 AWS Organizations していて、組織内で共有が有効になっている場合、組織内のコンシューマーには共有 Dedicated Host へのアクセスが自動的に付与されます。それ以外の場合、コンシューマーはリソース共有への参加の招待を受け取り、その招待を受け入れた後で、共有 Dedicated Host へのアクセス許可が付与されます。
- Windows Server 2019 は BYOL を実行できる最新バージョンであるため、ライセンス込みモデルで専有ホスト上で Windows Server 2022 を実行できます。専有ホストで Windows Server 2022

を使用する場合は、Windows Server 2022 ライセンス込みインスタンスを使用する必要があります。

- AWS License Manager は、 AWS およびオンプレミス環境のさまざまなベンダーからのソフトウェアライセンスを管理するための包括的なソリューションです。License Manager を使用すると、ソフトウェアライセンスの使用方法をより詳細に可視化し、制御できるため、コスト削減とコンプライアンスの向上につながります。License Manager を使用して、独自のライセンス条件をエミュレートするルールを設定できます。これにより、これらのルールを適用し、ライセンスの誤用を防ぐことができます。また、コンプライアンス違反のリスクが軽減され、ライセンス管理プロセスが向上します。
- License Manager を使用すると、ホストリソースグループを使用してホストの配置、リリース、復旧を自動化できます。これにより、生産性が向上し、管理オーバーヘッドが削減されます。License Manager では、ライセンスルールに基づいて AWS およびオンプレミス環境全体のライセンス使用状況を一元的に表示できるため、組織全体のライセンスの増分購入、コンプライアンス、ベンダー監査を簡単に管理できます。さらに、License Manager は AWS Organizations および AWS Resource Access Manager (AWS RAM) と統合して、アカウントとリージョン間でライセンス設定を共有します。これにより、スケジュールに基づいて環境全体のレポートを作成し、ライセンスルールを 1 つの AWS アカウントで一元管理できます。最終的には、ガバナンスを改善し、複雑さを軽減できます。
- 1 つのリージョン内で専有ホストの高可用性を設計する場合は、本番環境に不可欠なワークロード用に最低 2 つのアベイラビリティーゾーンに最低 2 つの専有ホストが割り当てられていることを確認してください。詳細については、「Amazon EC2 Dedicated Hosts for Microsoft Windows on AWS」リファレンスデプロイを参照してください。
- 各インスタンスサイズで実行可能な上限のインスタンス数は、専有ホストインスタンスファミリーごとに異なります。詳細については、Amazon EC2 ドキュメントの「専有ホストの設定表」を参照してください。

AWS ライセンスオプション

ライセンスは、次の主要なカテゴリに分類されます。

- ライセンス込み – このライセンスオプションを使用すると、ライセンスをオンデマンドで購入して使用でき、使用した分に対してのみ料金が発生します。これは、ライセンスの使用に柔軟性を求め、前払いコストを回避したいユースケースに最適です。Windows Server、SQL Server、およびその他のさまざまな Microsoft 製品から選択できます。
- ライセンスモビリティを備えた BYOL 製品 – 既存のライセンスがあり、それらをクラウドで使用する場合は、このライセンスオプションで、Microsoft のライセンスモビリティプログラムを通じ

て独自のライセンスをクラウドに持ち込むことができます。ソフトウェアアシュアランス (SA) 付きの SQL Server など、ライセンスモビリティを備えた製品は、共有テナンシーまたは専用テナンシーに持ち込むことができます。これにより、 AWS インスタンスのコストが削減されます。

- ライセンスモビリティのない BYOL 製品 – ライセンスモビリティのない Windows Server などの Microsoft 製品の場合、 はクラウドでこれらの製品を使用するための専用オプション AWS を提供します。さらに、専有ホストは物理コアレベルでライセンスを有効にし、ワークロードの実行に必要なライセンスを 50% 以上節約できます。専有ホストは、ほぼ常時実行される安定した予測可能なワークロードに最適な選択です。

Windows Server ライセンスの持ち込み

独自の Windows ライセンスの導入は、既存の投資を活用して AWS コストを削減できるため、ライセンス最適化の最も効果的な戦略の 1 つです。特定の BYOL シナリオでは、SA やライセンスモビリティのメリットは必要ありませんが、Amazon EC2 専用インフラストラクチャは常に必要です。対象となるには、2019 年 10 月 1 日より前に永久ライセンスを購入しているか、2019 年 10 月 1 日より前に有効であったエンタープライズ登録において調整として追加している必要があります。これらの特定の BYOL シナリオでは、2019 年 10 月 1 日より前に利用可能であったバージョンへのライセンスのみアップグレードできます。例えば、2017 年に SA を終了した場合は、Windows Server 2019 ではなく 2016 までのデプロイ権限しかありません。ただし、2019 は BYOL の対象となる最後のバージョンです AWS。詳細については、 AWS ドキュメントの「ライセンス – Windows Server」を参照してください。

ライセンスの持ち込みは、 AWSで Microsoft ワークロードを実行するコストに大きな影響を与える可能性があります。独自のライセンスを持ち込む場合、クラウドで実行されているインスタンスに対して追加のライセンスコストを支払う必要がないため、大幅なコスト削減につながる可能性があります。

次の表は、さまざまな設定で単一の c5.xlarge インスタンスを 24 時間 365 日実行した場合のオンデマンドの月額コストを示しています。

設定	月額コスト (USD)
Windows Server + SQL Server Enterprise Edition	1,353.00 USD (LI)
Windows Server + SQL Server Standard Edition	609.00 USD (LI)

設定	月額コスト (USD)
Windows Server のみ	259.00 USD (LI)
コンピューティングのみ (Linux)	127.00 USD

既存のライセンスを使用すると、ライセンスコストを削減し、全体的な AWS 請求額を節約できます。

Amazon EC2 専有ホストで BYOL の対象となるには、Windows Server や SQL Server などの独自のソフトウェアライセンスを持ち込む必要があります。BYOL を使用すると、 で既存のライセンスを使用でき AWS 、コスト削減につながります。独自のライセンスを持ち込むには、ソフトウェアベンダーから付与されたライセンス使用権限があり、ソフトウェアのインストールメディアまたはイメージも提供する必要があります。インストールメディアまたはイメージを使用して、専有ホストでインスタンスを起動できます。BYOL AMI の作成の詳細については、 AWS ブログの「Microsoft Workloads」の「How to create Windows Server Bring-Your-Own-License AMIs from on-premises with VM Import/Export AMIs」を参照してください。

> **Note**
>
> [自動] に設定されたライセンスタイプは、AWS ライセンス込みオプションと同等です。このオプションを使用すると、不要なオンデマンド支出が発生する可能性があります。ライセンスタイプを切り替える必要があります。

コスト最適化シナリオ

ライセンスの適切なサイズ設定と最適化は、 AWSでのコスト最適化の重要な要素です。適切な戦略を実装すれば、Amazon EC2 専有ホストと BYOL オプションを使用することで、ライセンスコストを削減し、コンプライアンスを維持し、ライセンス投資から最良の価値を実現することができます。

このセクションでは、次のようなシナリオ例について説明します。

- T3 専有ホストによるコスト削減
- SQL Server BYOL を使用した共有テナンシーと専有ホストの比較
- 高可用性 SQL Server のデプロイ

T3 専有ホストによるコスト削減

T3 専有ホストは、従来より固定 CPU リソースを提供する他の Amazon EC2 専有ホストとは異なります。T3 専有ホストは、対照的に、CPU リソースを共有し、ベースライン CPU パフォーマンスを提供し、必要に応じてバーストすることができる、バースト可能なインスタンスをサポートします。CPU リソースの共有はオーバーサブスクリプションとも呼ばれ、単一の T3 専有ホストで、同等の汎用専有ホストより最大 4 倍多くのインスタンスをサポートできるようになります。

T3 専有ホストは、他の Amazon EC2 専有ホストよりも高いインスタンス密度を提供することで、TCO を削減します。バースト可能な T3 インスタンスを使用すると、これまでよりも少ないホストで、低から中程度の平均 CPU 使用率を持つより多くのインスタンスを統合することができます。また、T3 専有ホストは、他の Amazon EC2 専有ホストよりも多くの vCPU とメモリの組み合わせで、より小さいインスタンスサイズを提供します。インスタンスサイズを小さくすると、TCO が低くなり、オンプレミスホストと同等以上の統合率を実現できます。

T3 専有ホストは、Microsoft Windows デスクトップ、Windows Server、SQL Server、Oracle Database など、低から中程度の CPU 使用率で、ソケット単位、コア単位、または VM 単位のソフトウェアライセンスが有効な BYOL ソフトウェアの実行に最適です。

T3 専有ホストを使用して Windows Server Datacenter ライセンス (コアあたり) を削減する

オンプレミス環境では、VMware ホストの物理 CPU を簡単にオーバーサブスクライブし、高レベルの統合を実現できるという事実を活用しています。

次の例を考えます。現在、オンプレミス環境で 10x36 コア、384 GB RAM VMware ホストを使用しています。さらに、各ホストは、平均 CPU 使用率が低い 96x2 vCPU、4 GB RAM Windows Server 仮想マシンを実行しています。

仮想マシンを T3 専有ホストに移動することで、より高いレベルの統合を実現できるようになりました。T3 専有ホストは、現在のオンプレミス VMware ホストの 2 倍の RAM 量を備えています。T3 専有ホストで同じ数のサーバーを 50% 少ないホストコストで実行できます。これにより、Windows Server のライセンスコストを 33% 削減できます。次の表は、T3 専有ホストの使用による節約を示しています。

	オンプレミスの VMware ホスト	T3 専有ホスト	削減量
物理サーバー	10	5	

	オンプレミスの VMware ホスト	T3 専有ホスト	削減量
ホストあたりの物理コア	36	48	
ホストあたりの RAM (GB)	384	768	
ホストあたりの 2 vCPU、4 GB RAM VM	96	192	
VM の総数	960	960	
Windows Server Datacenter ライセンスの合計 (コアあたり) = (サーバー数 * 物理コア数)	10 * 36 = 360	5 * 48 = 240	33%

SQL Server BYOL を使用した共有テナンシーと専有ホストの比較

Amazon EC2 専有ホストの価値を示す実用的な例について考えてみましょう。このシナリオでは、組織は 240 コアのオンプレミス環境で SQL Server ワークロードを実行し、同じワークロードを費用対効果の高い方法で AWSにデプロイしたいと考えています。この組織が独自のライセンスを持ち込む (BYOL) 場合、SA の支払いは継続し、コアの数を減らすとコストに直接影響します。

次の図は、Microsoft の使用権限と SQL Server の AWS 節約額を比較したものです。

Microsoft entitlements (Enterprise Agreements)	
	Number of cores
SQL Server Enterprise edition	208
SQL Server Standard edition	32
Total SA cost	**$341,000**

SQL Server savings with AWS	
AWS shared vCPUs	AWS BYOL/Dedicated Hosts cores
120	96
20	-
$197,418	**$151,355**

AWS 共有テナンシーでインスタンスのサイズを適切に設定することで、SQL Server ライセンスを 140 コアに減らすことができます。これにより、SA コストは 197,000 USD になります。

Amazon EC2 専有ホストを使用すると、物理コアレベルで SQL Server をライセンスできます。これは、SQL Server ライセンスがインスタンスに割り当てられた vCPU の数に基づいている共有テナンシーでは不可能です。したがって、それぞれ 48 コアを持つ 2 つの R5 専有ホストを使用することで、共有テナンシーで必要な 140 vCPU ではなく 96 コアだけをカバーすれば済みます。R5 専有ホストをデプロイし、ワークロードを物理レベルでライセンスすることで、必要な SQL Server Enterprise Edition ライセンスの数を 96 コアに減らすことができます。つまり、ライセンス要件を満たし、大幅なコスト削減を実現しながら、192 コア (ハイパースレッディングを考慮) の SQL Server ワークロードをデプロイできます。

この場合、組織は年間約 341,000 USD の SA コストを支払います。共有テナンシーで適切なサイズ設定を行った後、140 vCPU でコストを 197,000 USD に削減しました。Amazon EC2 専有ホストは、コストをさらに 151,000 USD に削減します (約 56% の削減)。

高可用性 SQL Server のデプロイ

この例では、さまざまなライセンスの考慮事項に基づいて、 AWS での SQL Server のデプロイにコストがどのように影響するかを分析します。組織が 3 つのアプリケーションをサポートするために 6 つの SQL Server Enterprise サーバー AWS を にデプロイする必要があるとします。これらのサーバーには高可用性が必要で、それぞれ 16 vCPU と 256 GB の RAM があります。次のシナリオの詳細を参照してください。

- サーバー – SQL Server
- オペレーティングシステムのエディション – Windows Server Datacenter 2019
- SQL Server のエディション – SQL Server Enterprise 2019
- vCPU – 16
- メモリ (GB) – 256
- 数量 – 6

パフォーマンスを犠牲に AWS せずに のコストを最適化するには、CPU、メモリ、ネットワーク、ディスク (IOPS/BW) 使用率に基づいてインスタンスのサイズを適切に設定することをお勧めします。ワークロードのサイズを適切に設定したら、16 vCPU を提供する x2iedn.4xlarge インスタンスタイプに配置します。ただし、このインスタンスタイプには、ワークロードに必要なメモリが 2 倍含まれています。さらなる最適化はまだ可能です。

シナリオ 1

組織は、Windows と SQL Server の両方のライセンス込みオプションを使用して、 AWS 共有テナンシーに 6 つの SQL Server Enterprise サーバーをデプロイします。このオプションでは、Windows

および SQL Server ライセンスのコストがインスタンス料金に組み込まれます。次のシナリオの詳細を参照してください。

- 共有テナンシー (インスタンス) – x2iedn.4xlarge
- 時間単位のコスト (USD) – 10.0705 USD
- ユニットあたりの月額コスト (USD) – 7,351.47 USD
- サーバーの数 – 6
- CPU – 16
- メモリ – 512
- 6 台のサーバーの月額コスト – 44,108 USD

シナリオ 2

組織には、共有テナンシー上の SQL Server に対する SA と BYOL があります。つまり、組織は Windows のライセンス込みオプションを使用しますが、インスタンスに割り当てられた vCPU の数に基づいて独自の SQL Server ライセンスを提供します。組織には 6 つの SQL Server Enterprise サーバーがあり、それぞれに 16 個の vCPU があるため、合計 96 vCPU が必要です。次のシナリオの詳細を参照してください。

- 共有テナンシー (インスタンス) – x2iedn.4xlarge
- 時間単位のコスト (USD) – 4.0705 USD
- ユニットあたりの月額コスト (USD) – 2971.47 USD
- サーバーの数 – 6
- CPU – 16
- メモリ – 512
- BYOL コア – 96
- 6 台のサーバーの月額コスト – 17,828 USD

SA 付きの独自の SQL Server ライセンスを持ち込むことで、このシナリオの組織は、SQL Server のライセンス込みオプションを使用する場合と比較してコスト削減を実現できます。正確なコスト削減は、特定のライセンス契約の料金と条件によって異なります。このシナリオでは、SQL Server Enterprise ライセンスを に持ち込むと、 AWS コストが 1 か月あたり 26,280 USD 減少します AWS。

シナリオ 3

組織には、Amazon EC2 専有ホスト上の Windows および SQL Server の両方に対する BYOL があります。つまり、組織は物理コアレベルでライセンスを割り当て、ホストの物理コアのみをライセンスできるようにします。物理コアレベルでのライセンスにより、必要なライセンスに影響を与えることなく、インスタンスの最大数をデプロイできます。このライセンスモデルは、Windows Server Datacenter および SQL Server Enterprise Edition で一般的に使用されます。

このシナリオでは、2 つの X2iezn Amazon EC2 専有ホストを使用します。各ホストには 24 個の物理コアと 48 個の vCPU があります。これにより、それぞれ 16 個の vCPU と 256 GB の RAM を備えた 6 個の SQL Server Enterprise サーバーに十分なキャパシティが提供されます。次のシナリオの詳細を参照してください。

- 専有ホストの数 – 2
- インスタンスファミリー – x2iezn
- 時間単位のコスト (USD) – 11.009 USD
- ユニットあたりの月額コスト (USD) – 8,036 USD
- 物理コア – 48
- 使用可能な vCPU – 96
- 必要な Windows Server コアライセンス – 24
- SQL Server Enterprise コアに必要なライセンス – 24
- 月額コスト – 16,073

2 つの X2iezn ファミリー Amazon EC2 専有ホストの合計コストは、1 か月あたり 16,073 USD です。料金の詳細については、このシナリオの AWS 料金見積りツール 見積り を参照してください。このシナリオの組織は、Windows ライセンスを持ち込むことで、1 か月あたり 1,755.65 USD 節約できます。Amazon EC2 専有ホストを使用する場合は、必要な SQL Server ライセンスの数を減らすこともできます。共有テナンシーでは、それぞれ 16 個の vCPU を持つ 6 個の SQL Server Enterprise サーバーをカバーするために、96 個の SQL Server Enterprise ライセンスが必要になります。ただし、Amazon EC2 専有ホストを使用し、物理コアレベルでライセンスすることで、必要なライセンスの数を 48 コアに減らすことができます。

次の詳細では、例 3 のコストを比較し、BYOL オプションを使用して Amazon EC2 専有ホストにワークロードをデプロイすることでどれだけ節約できるかを他のシナリオと比較して示します。

- オンプレミスサーバー – SQL Server

- vCPU – 16
- メモリ – 256
- サーバーの数 – 6
- シナリオ 1 の月額コスト: Windows (LI) + SQL Server Enterprise (LI) – 44,108 USD
- シナリオ 2 の月額コスト: Windows (LI) + SQL Server Enterprise (BYOL) – 17,828 USD
- シナリオ 3 の月額コスト: Amazon EC2 専有ホストでの Windows (LI) + SQL Server Enterprise (BYOL) – 16,073 USD

> **Note**
>
> コストはオンデマンド料金に基づいています。Savings Plans または専有リザーブドインスタンスを使用することで、さらにコストを削減できます。これらのオプションは、オンデマンド料金と比較して大幅なコスト削減を実現する柔軟な料金モデルを提供します。これらのプランでは、1 年または 3 年の期間をコミットできます。詳細については、このガイドの「Amazon EC2 での Windows の支出を最適化する」セクションを参照してください。

Amazon EC2 専有ホストでは、次の支払いオプションを検討してください。

- Dedicated Hosts (Amazon EC2 ドキュメント)
- Dedicated Host Reservations (Amazon EC2 ドキュメント)
- Savings Plans (Amazon EC2 ドキュメント)

AWS 料金見積りツール が専有ホストの料金をサポートするようになりました。これにより、基盤となる適切な専有ホストを選択できます。

コスト最適化の推奨事項

AWS Cost Explorerを使用してコストを最適化するには、次のステップを実行することをお勧めします。

1. Cost Explorer を有効にする。
2. Cost Explorer を使用して、Amazon EC2 専有ホストのデプロイのコストと使用状況を表示および分析します。

3. BYOL を実行していることを検証します。次のプラットフォームの詳細と使用オペレーションの値は、Amazon EC2 コンソールの [インスタンス] ページまたは [AMI] ページ、あるいは `describe-images` コマンドまたは `describe-instances` コマンドによって返されるレスポンスに表示できます。
   - プラットフォームの詳細: Windows、使用オペレーション: RunInstances:0002 (ライセンス込み)
   - プラットフォームの詳細: Windows BYOL、使用オペレーション: RunInstances:0800

その他のリソース

- 「Eligible license types for license type conversion」(AWS License Manager ドキュメント)
- AWS License Manager &専有ホストワークショップ (AWS License Manager ワークショップ)
- 「Amazon EC2 Dedicated Hosts FAQs」(AWS ドキュメント)
- 「How to create Windows Server Bring-Your-Own-License AMIs from on-premises with VM Import/Export」(Microsoft Workloads on AWS ブログ)
- VM Import/Export (AWS ドキュメント)
- アマゾン ウェブ サービスと Microsoft: よくある質問 (AWS ドキュメント)
- License Manager でのライセンスタイプの変換 (AWS License Manager ドキュメント)
- 高可用性 SQL Server を Amazon EC2 専有ホストにデプロイする (AWS クラウドオペレーションと移行ブログ)

Amazon EC2 での Windows の支出を最適化する

概要:

サーバーを に移行することに関する最大の懸念事項の 1 つは、インフラストラクチャコスト AWS です。確かにクラウドの利点の 1 つはリソースに対してオンデマンドで料金を支払うことですが、24 時間 365 日利用可能である必要がある本番ワークロードも存在します。Savings Plans は、EC2 インスタンス、 AWS Lambda、および 全体での定常状態 AWS の使用コストを削減するように設計されています AWS Fargate。

Savings Plans は柔軟な料金モデルを提供し、Amazon EC2、Fargate、Lambda、Amazon SageMaker AI の使用料を、一定の使用量 (1 時間あたり 10 USD など) のコミットメントと引き換えに割引するのに役立ちます。1 年または 3 年間にわたって一定量の時間単位のコンピューティング支出をコミットし、引き換えにその使用量に対して割引を受けます。

Savings Plans では、3 つの異なる支払いオプションから選択できます。

- [前払いなし] オプションでは、前払いを必要とせず、コミットメントは純粋に月単位で請求されます。
- [一部前払い] オプションでは、Savings Plans の料金が安くなります。コミットメントの少なくとも半分が前払いで請求され、残りは毎月請求されます。
- [すべて前払い] オプションでは、最低価格が提供され、コミットメント全体が 1 回の支払いで請求されます。

AWS Cost Explorerでは、現在アクティブな Savings Plans の有効期限と、キューに登録されている開始予定の Savings Plans を追跡することができます。Savings Plans アラートを使用すると、Savings Plan の有効期限の 1 日前、7 日前、30 日前、60 日前、またはコミットメントが購入用にキューに登録されているときに、事前のメールアラートを受け取ることができます。これらの通知は、有効期限についても警告します。最大 10 人の E メール受信者に通知を送信できます。

Savings Plans について

すべてのタイプのコンピューティング使用量には、オンデマンド料金と Savings Plans 料金があります。1 時間あたり 10 USD のコンピューティング使用量をコミットすると、Savings Plans 料金で最大 10 USD までのすべての使用量に対して Savings Plans 料金が適用されます。コンピューティング支出コミットメントを超えた使用については、通常のオンデマンド料金で請求されます。Savings Plans の使用を開始するには、 AWS マネジメントコンソールの Cost Explorer を使用します。

Cost Explorer で提供されている推奨事項を使用して最大のコスト削減を実現することで、Savings Plans へのコミットメントを簡単に行うことができます。推奨される時間単位のコミットメントは、過去のオンデマンド使用量と、プランタイプ、期間、支払いオプションの選択に基づいています。Savings Plans は、最初に、プランを購入したアカウントに適用され、次に一括請求ファミリー内の他のアカウントと共有されます。

> **Note**
>
> の Savings Plans 共有オプション AWS Organizations はデフォルトで有効になっています。このオプションは、支払者アカウントの AWS Billing コンソールで拒否できます。レコメンデーションページにアクセスして、対象となる使用量を節約するために が AWS 推奨する Savings Plans を確認できます。これらの推奨事項はいつでも更新できるため、最適な Savings Plans を簡単に購入できます。

Compute Savings Plans

Compute Savings Plans は、柔軟性が最も高く、コストの削減に役立ちます。これらのプランは、インスタンスファミリー、サイズ、アベイラビリティーゾーン、リージョン、オペレーティングシステム、またはテナンシーに関係なく、EC2 インスタンスの使用に自動的に適用されます。Fargate または Lambda の使用にも適用されます。例えば、Compute Savings Plans では、C4 インスタンスから M5 インスタンスへの変更、EU (アイルランド) から EU (ロンドン) へのワークロードの移行、EC2 から Fargate または Lambda へのワークロードの移動をいつでも行うことができます。Savings Plans の料金の支払いは自動的に引き続き行われます。

EC2 Instance Savings Plans

EC2 Instance Savings Plans は、リージョン内の個々のインスタンスファミリーの使用量に対するコミットメント (例えば、バージニア北部で一定のレベルの M5 使用量をコミットするなど) と引き換えに、最も大きい割引を提供します。これにより、アベイラビリティーゾーン、サイズ、オペレーティングシステム、テナンシーに関係なく、そのリージョンで選択したインスタンスファミリーのオンデマンド料金が自動的に割引されます。EC2 Instance Savings Plans では、そのリージョンのファミリー内のインスタンス間で使用量を変更できます。例えば、Windows を実行している c5.xlarge から、Linux を実行している c5.2xlarge に移行でき、自動的に Savings Plans 価格のメリットを受けることができます。

Compute Savings Plans と EC2 Instance Savings Plans はどちらも、Amazon EMR、Amazon Elastic Kubernetes Service (Amazon EKS)、および Amazon Elastic Container Service (Amazon ECS) クラスターの一部である EC2 インスタンスに適用されます。Amazon EMR、Amazon EKS、および Amazon ECS の料金は Savings Plans の対象ではありませんが、基盤となる EC2 インスタンスは対象です。Compute Savings Plans の方が適用範囲が広いため、EC2 Instance Savings Plans は Compute Savings Plans よりも優先して適用されます。

> **Note**
>
> Savings Plans は、コミット完了後に簡単に変更することはできません。いずれかの Savings Plans オプションをコミットする前に、慎重に計画することをお勧めします。Savings Plans は、コミットメントと引き換えにオンデマンド価格よりも低価格で提供されるもので、期間中にキャンセルできません。

時間単位のコミットメントの例

Savings Plans を購入する場合は、プランの期間に対して時間単位の金額的コミットメントを行います。1 時間あたり 10 USD のコンピューティング使用量をコミットすると、Savings Plans の料金は、1 時間あたり 10 USD までのすべての使用量に自動的に適用されます。コミット額を超えた使用については、通常のオンデマンド料金で請求されます。Cost Explorer の Savings Plans 購入推奨事項ツールを使用して、削減額を最大化できる推奨コミットメントを取得できます。特定のプランの時間単位の金額的コミットメントは、プランの期間中は変更できません。使用量の分析後にコミットメントを増やす場合は、超過使用量をカバーするために追加の Savings Plans を購入することができます。

Savings Plans の利点

リザーブドインスタンスと比較して、Savings Plans は、Savings Plans が提供する幅広いコンピューティングオプションを利用しながらコストを削減できる、より柔軟な料金モデルを提供します。Savings Plans は、コンピューティングのニーズが変化しても割引を提供します。これにより、追加の管理オーバーヘッドを発生させることなく、絶えず変化する動的環境に対応することができます。Savings Plans を使用するその他の利点は次のとおりです。

- 使いやすい – 金額的コミットメントと引き換えに、自動割引を受けられます。
- 柔軟性 – 複数の使用タイプに適用される単一のコミットメント。
- 潜在的な節約 – 節約にはさまざまな方法があります。次の例を考えます。
  - Compute Savings Plans を使用した Windows Server ワークロードの 60% の節約 (d2.8xlarge、3 年、すべて前払い、Windows、共有テナンシー、us-east-2)
  - EC2 Instance Savings Plans を使用した Windows Server ワークロードの 73% の節約 (d2.8xlarge、3 年、すべて前払い、Windows、共有テナンシー、us-east-2)
  - 非エキゾチックインスタンスタイプの 28～41% の節約 (t3 ファミリー、3 年、すべて前払い、Windows、共有テナンシー、us-east-2)
  - Windows Server の場合、平均 25～40% の節約

> **Note**
>
> EC2 Instance Savings Plans は、柔軟性が低いため、Compute Savings Plans よりも大きな割引を提供します。割引料金の使用量をコミットします。

すべてのタイプのコンピューティング使用量には、Savings Plans 料金とオンデマンド料金があります。次の表は、オペレーティングシステムタイプごとの Savings Plans 料金とオンデマンド料金を示しています。コミットされた使用量に対しては Savings Plans 料金で請求され、コミットメントを超えた使用量に対しては通常のオンデマンド料金で請求されます。

Instance name	Savings Plans 料金	オンデマンドの節約	オンデマンド料金	オペレーティングシステム	リージョン	お支払い方法	期間
x2iedn.xlarge	0.32 USD	61%	0.83 USD	Linux	米国東部 (バージニア北部)	前払いなし	3
x2iedn.xlarge	2.01 USD	50%	1.02 USD	Server	米国東部 (バージニア北部)	前払いなし	3
x2iedn.xlarge	1.02 USD	20%	2.52 USD	Windows ライセンス込み + SQL Server Enterprise Edition	米国東部 (バージニア北部)	前払いなし	3
x2iedn.xlarge	0.32 USD	61%	0.83 USD	BYOL	米国東部 (バージニア北部)	前払いなし	3

Savings Plans にはオペレーティングシステムが含まれており、BYOL には別の割引が適用されます。これらはすべて Compute Savings Plans 計算ツールで内訳が示されています。

リザーブドインスタンス料金モデル

AWS には、リザーブドインスタンスと呼ばれるコミットメントに基づく別の料金モデルがあります。このモデルは、既にコミットメントを行った後にコンピューティングが変更され、リザーブドインスタンスが使用できなくなると、問題が発生する可能性があります。Savings Plans は、スタンダードリザーブドインスタンスやコンバーティブルリザーブドインスタンスと同様のコスト削減を実

現するように設計されていますが、柔軟性ははるかに高くなります。Compute Savings Plans では、インスタンスファミリー、サイズ、オペレーティングシステム、テナンシー、 リージョンに関係なく、EC2 インスタンスの使用料金を削減できます。また、最大限の柔軟性も実現します。

次の表は、Savings Plans またはリザーブドインスタンスの選択に役立ちます。

	Reserved Instance	EC2 Instance Savings Plans	Compute Savings Plans
1 年間の平均割引	最大 38%	最大 29%	最大 29%
3 年間の平均割引	最大 58%	最大 73%	最大 60%
インスタンスファミリー	[固定]	[固定]	フレキシブル
インスタンスサイズ	固定 (Linux 以外)	フレキシブル	フレキシブル
地域別	1 リージョン	1 リージョン	フレキシブル
オペレーティングシステム	[固定]	フレキシブル	フレキシブル
サービス	Amazon EC2 または Amazon RDS	Amazon EC2	Amazon EC2、Fargate、Lambda
支払いオプション	すべて前払い、一部前払い、前払いなし	すべて前払い、一部前払い、前払いなし	すべて前払い、一部前払い、前払いなし
インスタンス制限	アベイラビリティーゾーンあたり 20	無制限	無制限

> **Note**
>
> Savings Plans は、時間単位の金額的コミットメントに基づいて割引を提供することで機能します。時間単位の金額的コミットメントは、プランの期間中はキャンセルも変更もできませんが、追加の使用量をカバーするために追加の Savings Plans を購入できます。これにより、フリートの増加に合わせて一定の時間単位のコミットメントを維持できます。

AWS Cost Explorer や AWS クラウド インテリジェンスダッシュボードなどのツールを使用して、コミットメントを追跡できます。Cost Explorer は、組織が Savings Plans カバレッジ戦略を計画するのに役立つカバレッジターゲット線を提供します。ワークロードの 75% が定常状態の場合は、75% が適切なターゲットです。これにより、支出の 25% が、動的ワークロードに基づいてオンデマンド/可変となります。これを 85% のカバレッジまで増やす必要がある場合は、Savings Plans のコミットメントをもう 1 つ購入して、時間単位の金額的コミットメントを増やすことができます。

> **Note**
>
> リザーブドインスタンスの代わりに Savings Plans を購入することをお勧めしますが、リザーブドインスタンスを既に購入している場合は、2 つのコミットメントモデルを連携させることができます。

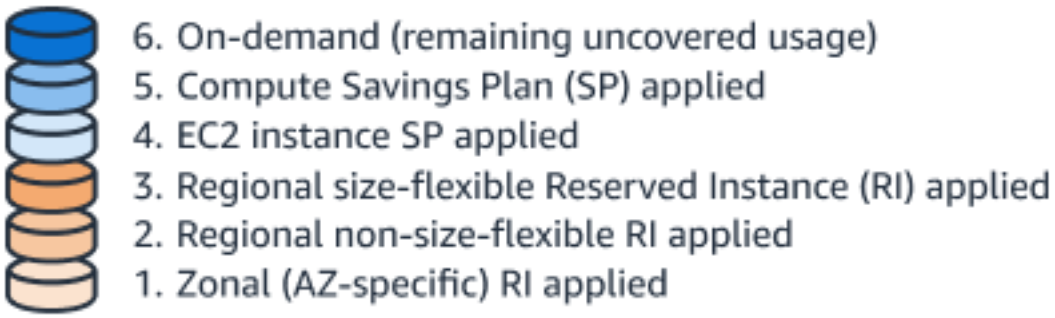
リザーブドインスタンスを購入したが、Savings Plans オプションの試用を開始する例について考えてみましょう。この組み合わせを最終請求に適用するロジックがあります。 AWS アカウントに適用できる階層を次に示します。

1. ゾーンリザーブドインスタンスは、それを所有するアカウントに適用されます。リザーブドインスタンスに残り時間がある場合は、組織の残りの部分に適用されます。
2. Windows 用のサイズ柔軟ではないリージョンリザーブドインスタンスは、それを所有するアカウントの対応する使用量に適用されます。残っているものは、組織の残りの部分にロールアウトされます。
3. サイズ柔軟なリージョンリザーブドインスタンスは、それを所有するアカウントに適用されてから (ファミリー内の最小インスタンスから順に大きいインスタンスへ)、組織の残りの部分に適用されます。
4. リージョンリザーブドインスタンスは、未使用のオンデマンドキャパシティ予約に適用されます。
5. EC2 Instance Savings Plans は、それを購入したアカウント内に適用されます。
6. Compute Savings Plans は、それを購入したアカウント内に適用されます。

> **Note**
>
> 割引は、最も大きい割引が適用される使用量から始まり、最も小さい割引へと進んでいきます。Windows インスタンスは、従来、最も一般的なインスタンスタイプ (T3、M6、C5 など)

では Linux よりも割引の可能性が低くなっています。つまり、ほとんどの場合、Linux インスタンスは Windows インスタンスよりも多くの利点があります。

次の図は、リザーブドインスタンスを Savings Plans から分けた後の料金を示しています。Compute Savings Plans と EC2 Instance Savings Plans の両方が、まず実行中のインスタンスに適用され、次に未使用のオンデマンドキャパシティ予約に適用されます。



コスト最適化シナリオ

このセクションでは、ライセンス込み請求モデルを使用する Amazon EC2 専有ホストおよび Amazon EC2 インスタンスのコスト最適化シナリオについて説明します。

Amazon EC2 専有ホスト

オンプレミスの Windows ワークロードを AWSに移行するシナリオを考えてみましょう。データセンターには次のサーバーがあります。

- 16 個の vCPU と 128 GB RAM を搭載した 2 台のサーバー
- 32 個の vCPU と 164 GB RAM を搭載した 2 台のサーバー
- 8 個の vCPU と 64 GB RAM を搭載した 1 台のサーバー
- vCPU と 32 GB RAM を搭載した 16 台のサーバー

さらに、引き継ぐのに十分なライセンスがあるため、独自のライセンスを AWS に持ち込むことができると仮定します。次の表は、 AWSで使用できるサーバーインスタンスを示しています。

インスタンスタイプ	CPU	RAM	Amount
r5.4xlarge	16	128	2

インスタンスタイプ	CPU	RAM	Amount
r5.8xlarge	32	256	2
r5.2xlarge	8	64	1
r5.xlarge	4	32	16
			21

分析では、これらの 21 台の仮想マシンを、R5 インスタンスファミリーホストを持つ 2 つの専有ホストに分散できることが示されています。次の表は、これら 2 つの専有ホストのコストを示しています。

専有ホストオンデマンドシナリオ	前払い	1 か月	1 年	3 年	AWS 料金見積りツール
[オンデマンド]	なし	10,123 USD	121,475 USD	364,392 USD	AWS 料金見積りツール見積り
1 年間の Savings Plans	なし	7,447 USD	89,362 USD	–	AWS 料金見積りツール見積り
3 年間の Savings Plans	なし	5,476 USD	65,712 USD	197,128 USD	AWS 料金見積りツール見積り
前払いによる 3 年間の Savings Plans	84,438 USD	2,755 USD	117,499 USD	183,618 USD	AWS 料金見積りツール見積り

移行先のサーバーがある場合 AWS、1 年間の Savings Plans の最終料金は、オンデマンド料金で 121,475 USD ではなく、89,362 USD です。これは、1 年後には 26.5% の割引となるということで

す。 AWS より長期間の滞在を検討している場合は、より深いコスト削減のために 3 年間の Savings Plans を選択できます。3 年後には、364,392 USD ではなく 197,128 USD を支払います。これにより、3 年後には総額の 46% を節約できます。

ライセンス込みの Amazon EC2 インスタンス

単一の 3 層アプリケーションを に移行し AWS、 が提供するライセンスを使用するシナリオを考えてみましょう AWS。さらに、アプリケーションが次のサーバーで動作すると仮定します。

- 2 個の vCPU と 4 GB RAM を備えた 2 台のウェブサーバー
- 8 個の vCPU と 16 GB RAM を搭載した 2 台のアプリケーションサーバー
- 16 個の vCPU と 64 GB RAM を搭載した 2 台のデータベースサーバー (SQL Server Standard Edition を使用)

次の表は、 AWSで使用できるサーバーインスタンスを示しています。

インスタンスタイプ	CPU	RAM	Amount
c5.large	2	4	2
c5.2xlarge	8	16	2
r5.2xlarge	8	64	2
			6 サーバー

次の表は、 のこれらのサーバーのコストを示しています AWS。

によって含まれるライセンス AWS	前払い	1 か月	1 年	3 年	AWS 料金見積りツール
[オンデマンド]	なし	3,912 USD	46,950 USD	140,849 USD	AWS 料金見積りツール見積り

によって含まれるライセンス AWS	前払い	1 か月	1 年	3 年	AWS 料金見積りツール
1 年間の Savings Plans	なし	3,466 USD	41,952 USD		AWS 料金見積りツール見積り
前払いなしの 3 年間の Savings Plans	なし	3,189 USD	38,264 USD	114,804 USD	AWS 料金見積りツール見積り
前払いによる 3 年間の Savings Plans	112,110 USD	なし	なし	なし	AWS 料金見積りツール見積り

オンデマンド料金で、本番環境 (24 時間 365 日) でこれらのサーバーを実行する場合は、月額 3,912 USD を支払います。この月額コストの支払いは、1 年後には 46,950 USD、3 年後には合計 140,849 USD になります。

前払いなしで 1 年間の Savings Plans を選択した場合、月額コストは 3,466 USD に減少します。最初の年の終了時に、41,952 USD を支払います。これは合計 11% の割引です。前払いなしで 3 年間の Savings Plans を選択した場合、月額コストは 3,189 USD に減少します。3 年後には、114,804 USD を支払います。これにより、18.5% の節約になります。

コスト最適化の推奨事項

どちらのシナリオも、 AWSでワークロードを計画および予測するときにコストを削減するのに役立ちます。2 番目のシナリオの割引は、1 番目のシナリオと比較して少ないことを認識することが重要です。2 番目のシナリオでは、ライセンス料金はクラウドサーバーの料金に含まれています。 AWS はライセンス料金の割引を提供しませんが、常にライセンスを持ち込むことができ (特定のシナリオの場合) 、常に最適なコンピューティング/インスタンス料金を保証 AWS できます。

コンピューティングリソースとインスタンスリソースへの AWS 支出を制御するには、以下を実行することをお勧めします。

- 推奨事項にアクセスする
- 必要に応じて推奨事項をカスタマイズする
- 時間単位のコミットメントを確認する

推奨事項にアクセスする

Amazon EC2 コンソールを使用して、Savings Plans の推奨事項にアクセスできます。推奨事項は CSV 形式でダウンロードして、後で確認することもできます。詳細については、Savings Plans ドキュメントの「Monitoring your Savings Plans」を参照してください。

必要に応じて推奨事項をカスタマイズする

Amazon EC2 コンソールを開き、[インスタンス] セクションを展開して、[Savings Plans] を選択します。このページには、推奨事項を行う前後のインスタンスとコンピューティングの料金が表示されます。推奨事項では、次の要素を調整することもできます。

- 期間 – 1 ~ 3 年など
- 支払いオプション – 前払い、一部前払い、前払いなしなど
- 履歴 – 過去 7 日間、30 日間、60 日間など

時間単位のコミットメントを確認する

同じ例を使用して、24 時間 365 日稼働しているインスタンスがあるとします。推奨事項は、Savings Plans を使用することです。サイズに応じて、オンデマンド料金は 1 時間あたり 120 USD です。1 時間あたり 90 USD をコミットすることもできますが、これはリージョン、インスタンス、購入オプションによって異なる場合があります。この例では、オンデマンドコストと比較して 25% を節約できます。また、使用率とカバレッジが定義したしきい値を下回っているかどうかを追跡し、予算の終了が近づいた際のアラートを設定することもできます。

推奨事項を確認する

Savings Plans の推奨事項を慎重に確認することをお勧めします。 AWS はアクセス許可なしでは何も変更しません。これらは推奨事項のみであり、適用するかどうかはユーザーが決定します。

プランを購入する

Amazon EC2 コンソールを開き、[インスタンス] セクションを展開して、[Savings Plans] を選択します。次に、[Savings Plans の購入] を選択します。要件に基づいて、期間、リージョン、インスタ

ンスファミリー、時間単位のコミットメント、支払いオプション、さらには開始日まで選択できます。Compute Savings Plans、EC2 Instance Savings Plans、SageMaker AI Savings Plans から選択できます。詳細については、Savings Plans ドキュメントの「Purchasing Savings Plans」を参照してください。

使用率レポートを取得する

Savings Plans を購入すると、使用率レポートを取得できます。このレポートは、使用率を確認し、購入したプランが割引をカバーして最大化するのに十分かどうかを確認し、新しい割引をキャンセルまたは追加するのに役立ちます。このレポートは、CSV などの他の形式にエクスポートできます。詳細については、Savings Plans ドキュメントの「Using the utilization report」を参照してください。

購入のベストプラクティスに従う

Savings Plans を購入する前に、次のベストプラクティスに従うことをお勧めします。

- AWS Trusted Advisor を使用して、アイドル状態の EC2 リソースを削除します。
- Savings Plans の購入前に適切なサイズ設定を行います。
- 30～60 日間一貫して維持する時間単位の料金を設定します。
- 組織が許容できる範囲の、一貫した時間単位の料金をカバーするコミットメントを購入します。需要や季節の変動を考慮します。
- Savings Plans の予算を四半期ごとに見直して、一貫したレート (Savings Plans カバレッジの 70% のカバレッジターゲットなど) を維持します。レートが希望するカバレッジを下回った場合は、カバレッジ目標を達成するための調整として追加の Savings Plans を購入します。

その他のリソース

- Amazon EC2 リザーブドインスタンスの Savings Plans (AWS ホワイトペーパー)
- Savings Plans が AWS 使用状況にどのように適用されるかを理解する (Savings Plans ドキュメント)
- EC2 Windows Server および SQL Server インスタンスの 1 秒あたりの請求の発表 (AWS ドキュメント)
- AWS コスト最適化シリーズ: Savings Plans Video | Amazon Web Services (YouTube)

AWS ツールを使用したコストのモニタリング

概要:

コストの可視性は、 のコストを最適化するための重要な要素です AWS。 AWS には、コストを視覚化し、それらのコストに応じてアラートを作成するために使用できるツールが多数あります。これには、支出の追跡と報告 AWS Budgetsに役立つツールなどがあります。このセクションでは、 AWS 上の Windows の支出をモニタリングする特定の方法について説明し、予算要件に応じて追跡して対応できるようにします。これには、Windows EC2 リソースへの必要なタグの追加が含まれます。これらのタグを使用すると、 AWS Budgetsで Windows EC2 およびその他の Microsoft サービスを適切にモニタリングできるようになります。

支出をモニタリングし、 AWS ツールでアラートを作成することで、現在の支出、予測支出、支出の異常についてより詳細に把握できます。Savings Plans を使用して時間単位の EC2 インスタンス料金を削減する場合は、Savings Plans の全体的な使用率とカバレッジを確認することをお勧めします。これにより、コスト削減を継続的に実現できるようになります。 AWS Cost Explorer を使用して Savings Plans インベントリを表示し、以前の使用状況に基づいて追加の Savings Plans のレコメンデーションを取得できます。AWS Budgets を使用して AWS Cost Anomaly Detection を設定することで、特定の支出を追跡することもできます。

コスト最適化の推奨事項

Cost Explorer AWS Budgetsと異常検出を使用してコストを最適化するには、次のステップを実行することをお勧めします。

- Windows EC2 リソースにタグを付ける
- を使用してアラートを設定する AWS Budgets
- コスト異常検出を有効にする
- リアルタイムの支出分析を取得する
- Cost Explorer を使用して Windows のライセンス込み支出を表示する

Windows EC2 リソースにタグを付ける

AWS 支出を効果的にモニタリングするには、モニタリングするワークロードのタグ付け戦略を確立する必要があります。これは、一般的な使用量の支出とは対照的に、リソースをカテゴリ別にグループ化し、特定の支出について通知を受けるために重要です。コストに役立つだけでなく、AWS

Systems Manager 自動化などの他の目的にも使用できるタグ付けリソースを使用できます。さらに、必要なタグの管理を実装することをお勧めします。

Cost Explorer AWS Budgets、および Cost Anomaly Detection の支出を追跡するには、適切なタグが設定されていることを確認する必要があります。 Cost Explorer タグを使用して、それらのタグに一致する項目の特定の予算を設定し、支出の増加時にアラートを受け取ることができます。

例えば、Key=OS Value=Windows などのシンプルなタグを使用できます。これにより、すべての Windows インスタンスが 1 つのグループにまとめられ、支出を追跡できるようになります。Systems Manager などの他の項目にタグを使用することもできます。タグを作成したら、コスト追跡のためにタグをアクティブ化する必要があります。特定のリソースにアタッチされているAWS Config タグをモニタリングする ルールを追加することを検討してください。 は、適切なタグが含まれていないリソースが実行されている場合はアラート AWS Config を受け取ることができます。これにより、Windows EC2 の支出を正確に把握できます。

タグを設定したら、 AWS Billingでカスタム予算を作成できます。これにより、Windows EC2 の支出を可視化できます。日次予算または月次予算を設定できます。

を使用したアラートの設定 AWS Budgets

このシナリオ例では、Windows EC2 の日次予算を作成します。これは、自動調整オプションを使用して支出を追跡し、それに応じて予算を調整する定期予算です。静的環境がある場合は、代わりに固定予算を使用できます。必ずベースラインの時間範囲 (30 日など) を選択してください。

1. にサインイン AWS マネジメントコンソール し、 AWS Cost Management コンソールを開きます。
2. ナビゲーションペインで、[予算] を選択します。
3. ページの上部で、[予算を作成] を選択します。
4. [Budget setup] (予算の設定) で、[Customize (advanced)] (カスタマイズ (高度)) を選択します。
5. [予算タイプ] で、[コスト予算] を選択します。次に、[次へ] を選択します。
6. [詳細] の [予算名] で、予算の名前を入力します。例えば、Windows EC2 spend などです。
7. [予算額を設定] の [期間] で、[日次] を選択します。
8. [予算更新タイプ] では、予算期間後にリセットする予算の [定期予算] を選択します。
9. [開始日] で、予算金額に対する追跡を開始する開始日または期間を選択します。
10. [予算設定方法] で、[自動調整 (新機能)] を選択します。
11. [ベースラインの時間範囲] で、[カスタム範囲] を選択し、30 日と入力します。

12[次へ] を選択します。

13Budget scope セクションで、Filter specific AWS cost dimensions を選択します。ここでは、タグを使用して適切なディメンションを作成します。 AWS Budgets は、フィルターのオプションとしてプラットフォームタイプをサポートしていません。このため、[OS] タグを適用する必要があります。

14[フィルタを追加] を選択し、[ディメンション] から [タグ] オプションを選択します。

15[OS] タグを選択し、この Windows 値を選択してタグの予算を作成します。

16[次へ] を選択します。

17[アラートの設定] ページで、[アラートのしきい値を追加] を選択します。ここでは、2 つのアラートを設定します。1 つは 50% のしきい値、もう 1 つは 100% のしきい値です。月の中間ポイントより前に 50% のしきい値アラートを超えた場合、警告が表示されます。これにより、支出が予想を上回っているかどうかを確認し、月末に達する前に対応することができます。

18[しきい値] には 50 と入力し、[予算額の %] を選択します。

19[トリガー] で、[実際] を選択します。

20[E メールの受信者] に、E メールアドレスを入力します。もう 1 つ、しきい値 100 のアラートを追加します。

> **Note**
>
> この例では、アラートに E メール通知を使用しますが、Slack などの他のアプローチを使用することもできます。

コスト異常検出を有効にする

コストタグを使用して、異常の支出アラートを設定できます。例えば、AWS Cost Anomaly Detection を使用して支出のモニターを作成し、システムがアカウントの異常な支出を検出したときにアラートを受け取ることができます。

前に作成した Key=OS および Value=Windows タグのモニターとアラートを設定するには、以下を実行します。

1. にサインイン AWS マネジメントコンソール し、 AWS Cost Management コンソールを開きます。
2. ナビゲーションペインで、[コスト異常検出] を選択します。
3. [コストモニター] タブを選択し、[モニターを作成] を選択します。

4. ステップ 1 で、モニタータイプとして [コスト配分タグ] を選択します。
5. [コスト配分タグキー] で、[Windows EC2 の支出] を選択します。
6. [コスト配分タグ値] で、[Windows] を選択します。
7. [モニターに名前を付ける] で、Windows EC2 spend と入力します。
8. [次へ] を選択します。
9. アラートのサブスクリプションを作成するには、[新しいサブスクリプションの作成] を選択します。既存のサブスクリプションがある場合は、[既存のサブスクリプションを選択] を選択します。
10. [サブスクリプション名] に、Windows EC2 spend anomaly と入力します。
11. [アラート頻度] で、[日次の要約] を選択します。
12. [アラート受信者] に、E メールアドレスを入力します。
13. [しきい値の追加] を選択します。[しきい値] に 10 と入力し、[予想速度を上回った割合] を選択します。
14. [Create monitor] (モニターの作成) を選択します。

支出をリアルタイムで表示する

アラートは Windows EC2 の支出をモニタリングするための便利なツールですが、支出をリアルタイムで表示する場合は Cost Explorer を使用する必要があります。Cost Explorer を使用して EC2 コストを分析および削減する方法については、この動画をご覧ください。詳細については、YouTube の「AWS Supports You | Understanding and Reducing Your EC2 Costs」の動画をご覧ください。

Windows のライセンス込み支出を表示する

Cost Explorer を使用して、アカウントの EC2 Windows の支出を表示できます。Windows のライセンス込み支出を確認するには、Cost Explorer で次の適切なフィルタを設定する必要があります。

- [プラットフォーム] で、[Windows (Amazon VPC)] を選択します。[API オペレーション] で、[RunInstance:0002] を選択します。これは、ライセンス込みの Windows EC2 インスタンスの AWS Billing コードです。
- BYOL インスタンスの支出を表示する場合は、[RunInstance:0002] を [RunInstance:0800] に変更します。これは Windows EC2 BYOL の請求コードです。

Cost Explorer でこの可視性を使用すると、Windows EC2 に費やしているコストを正確にすばやく絞り込むことができます。 AWS 支出をさらに深く掘り下げたい場合は、 AWS Cost and Usage

Report を使用して個々のインスタンスレベルで支出に絞り込むことができます。Amazon Quick で視覚化できるレポートを生成し、カスタマイズされたダッシュボードを構築することもできます。

詳細については、YouTube の「AWS Supports You - Visualizing Your Cost and Usage Reports」の動画をご覧ください。

その他のリソース

- で必要なタグを設定する AWS Config (AWS Config ドキュメント)
- AWS Budgets チュートリアル - のアラートの設定 AWS Billing | アマゾン ウェブ サービス (YouTube)
- AWS Cost and Usage Report クエリライブラリ (AWS Well-Architected Labs)

SQL Server

お客様は、他のクラウドプロバイダーよりも 15 年以上 AWS 、 で Microsoft ワークロードを実行しています。これは主に、 AWS がクラウド内の Microsoft アプリケーションで最も豊富な経験を持ち、Windows Server と Microsoft SQL Server に最適なプラットフォームを次の点で提供するためです。

- より高いパフォーマンスおよび信頼性
- より優れたセキュリティおよびアイデンティティサービス
- さらなる移行サポート
- 最も広範で深い機能
- 低い総保有コスト (TCO)
- 柔軟なライセンスオプション

AWS は、Active Directory、.NET、SQL Server、サービスとしての Windows デスクトップ、サポートされているすべてのバージョンの Windows Server など、SQL Server に依存する Windows アプリケーションの構築と実行に必要なすべてをサポートします。実証済みの専門知識により、 AWS は Windows ワークロードを簡単にリフトアンドシフト、リファクタリング、またはモダナイズするのに役立ちます。

本ガイドのこのセクションでは、次のトピックについて説明します。

- 高可用性とディザスタリカバリのソリューションを選択する
- SQL Server ライセンスを理解する
- SQL Server ワークロードに適した EC2 インスタンスを選択する
- インスタンスを統合する
- SQL Server エディションの比較
- SQL Server Developer Edition を評価する
- Linux 上の SQL Server を評価する
- SQL Server バックアップ戦略を最適化する
- SQL Server データベースをモダナイズする
- SQL Server のストレージを最適化する
- Compute Optimizer を使用して SQL Server ライセンスを最適化する
- Compute Optimizer を使用して SQL Server のサイズ設定を最適化する

- SQL Server ワークロードの Trusted Advisor レコメンデーションを確認する

高可用性とディザスタリカバリのソリューションを選択する

概要

目標復旧時間 (RTO) や目標復旧時点 (RPO) などのディザスタリカバリ (DR) 目標も達成しながらビジネスニーズを満たす、 AWS 上の SQL Server デプロイ用のアーキテクチャを設計することをお勧めします。以下のソリューションは、SQL Server ワークロードのコストも最適化しながら、Amazon Elastic Compute Cloud (Amazon EC2) 上の SQL Server に適したアーキテクチャを設計するのに役立ちます。

- SQL Server Always On 可用性グループ – SQL Server Always On 可用性グループは、SQL Server データベースの高可用性とディザスタリカバリ (HA/DR) ソリューションを提供します。可用性グループは、フェイルオーバーされるユーザーデータベースのセットで構成されます。Always On 可用性グループもデータベースレベルで冗長性を提供しますが、共有ストレージは必要ありません。各レプリカには独自のローカルストレージがあります。この機能は HA/DR ソリューションとしてデプロイできます。詳細については、Microsoft ドキュメントの「What is an Always On availability group?」を参照してください。
- SQL Server Always On フェイルオーバークラスターインスタンス (FCI) – SQL Server Always On FCI は Windows Server フェイルオーバークラスタリング (WSFC) を使用して、SQL Server インスタンスレベルで HA を提供します。FCI では、データベースをホストするために共有ストレージが必要です。共有ブロックストレージまたは共有ファイルストレージを使用できます。例えば、Amazon FSx for Windows File Server または Amazon FSx for NetApp ONTAP を、複数のアベイラビリティーゾーンを持つ共有ストレージソリューションとして使用できます。詳細については、Microsoft ドキュメントの「Always On Failover Cluster Instances (SQL Server)」を参照してください。
- SIOS DataKeeper – SIOS DataKeeper は、アベイラビリティーゾーンと AWS リージョンの両方にまたがる SQL Server FCI を有効にすることで、HA と DR の両方の要件を満たすのに役立ちます。SIOS DataKeeper は、ローカル Amazon Elastic Block Store (Amazon EBS) ボリュームを使用してクラスター化された仮想 SAN を作成し、HA のアベイラビリティーゾーン間の同期レプリケーションを使用し、ディザスタリカバリのためにリージョン間の非同期レプリケーションを使用します。詳細については、SIOS ドキュメントの「High Availability Protection for Windows Applications」を参照してください。
- 分散可用性グループ – 分散可用性グループは、2 つの別々の Always On 可用性グループにまたがる特殊な種類の可用性グループです。可用性グループは、2 つの別々のリージョン (us-east-1

や us-west-1 など) にまたがって存在できます。基盤となる Always On 可用性グループは 2 つの異なる WSFC クラスターに設定されているため、分散可用性グループを、可用性グループの可用性グループと考えることができます。分散可用性グループをデプロイするには、SQL Server Enterprise Edition が必要です。詳細については、Microsoft ドキュメントの「Distributed availability groups」を参照してください。

- ログ配布 – 万が一リージョンが影響を受けて使用できなくなった場合に備えて、ログ配布を実装して、複数のリージョンにまたがるデータベースを保護できます。トランザクションとログ配布の頻度に応じて、数分以内に RPO と RTO を達成できます。詳細については、Microsoft ドキュメントの「About Log Shipping (SQL Server)」を参照してください。
- AWS Elastic Disaster Recovery – Elastic Disaster Recovery は、DR AWS 目的で、あらゆるインフラストラクチャから へのサーバーのレプリケーションを管理する Software as a Service (SaaS) アプリケーションです。Elastic Disaster Recovery を使用して、リージョン間で SQL Server をレプリケートすることもできます。Elastic Disaster Recovery は、オペレーティングシステム、インストールされているすべてのアプリケーション、すべてのデータベースを含む仮想マシン全体をステージングエリアにレプリケートするエージェントベースのソリューションです。詳細については、Elastic Disaster Recovery ドキュメントの「What is Elastic Disaster Recovery?」を参照してください。
- AWS Database Migration Service (AWS DMS) – は AWS、別のリージョンを含む、 との間のデータのライブ移行 AWS DMS をサポートします。この機能を使用して、ディザスタリカバリデータベースとして機能する別のリージョンに別個の SQL Server インスタンスをセットアップできます。詳細については、「 とは」を参照してください AWS Database Migration Service。 AWS DMS 「」を参照してください。

SQL Server Always On 可用性グループ

高可用性 Always On 可用性グループのみに SQL Server Enterprise Edition を使用している場合は、基本的な可用性グループを利用して SQL Server Standard Edition にダウングレードできます。Always On 可用性グループの代わりに基本的な可用性グループを使用することで、コストを 65 ~ 75% 削減できます。

> **Note**
>
> 異なる SQL Server エディション間のコストの違いの詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。

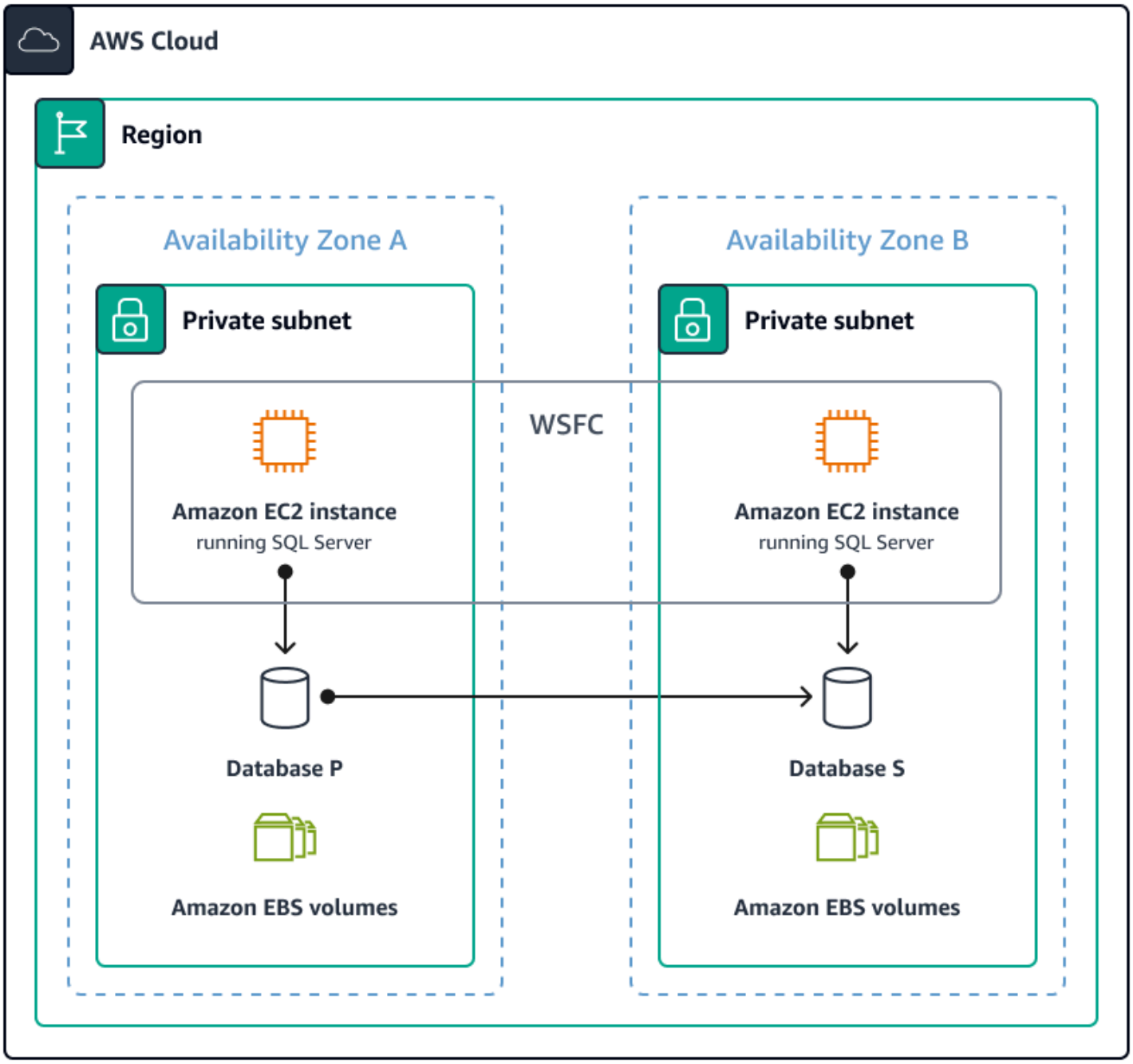
特徴

- SQL Server Standard Edition で使用可能
- レプリカは 2 つまで (プライマリとセカンダリ)
- セカンダリレプリカへの読み取りアクセスなし
- セカンダリレプリカの整合性チェックなし

制限事項

- 可用性グループごとに 1 つの可用性データベースのみをサポート
- 基本的な可用性グループを分散可用性グループの一部にすることはできない

次の図は、Windows Server フェイルオーバークラスターソリューションのアーキテクチャの例を示しています。



SQL Server Always On フェイルオーバークラスターインスタンス

フェイルオーバークラスターインスタンス (FCI) を使用して、ダウンタイムを最小限に抑え、データ損失のリスクを減らしながら、データベースの継続的なオペレーションを確保できます。FCI は、リードレプリカの設定なしで SQL Server データベースの高可用性を求める場合に、信頼性の高いソリューションを提供します。

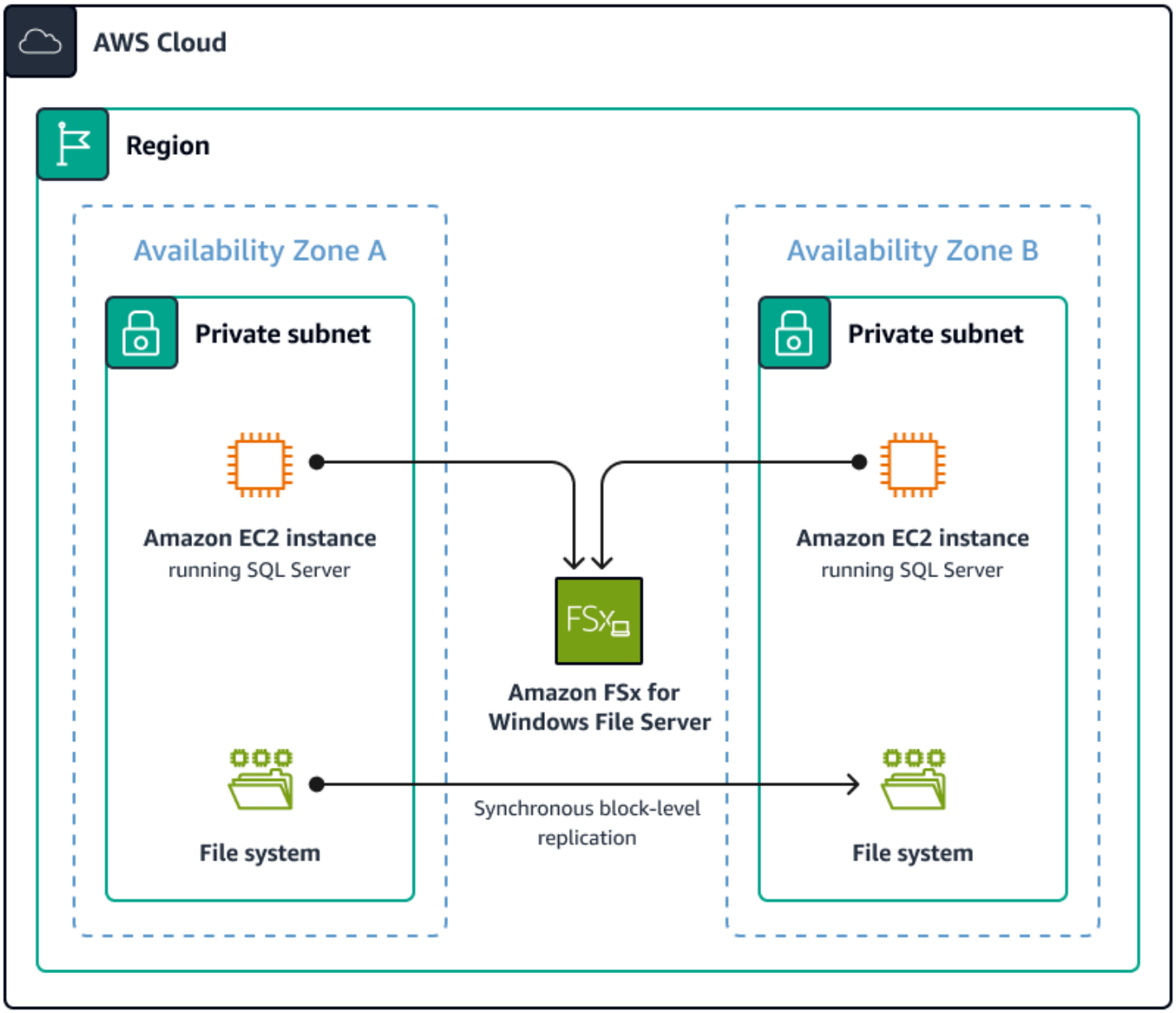
可用性グループとは異なり、FCI は SQL Server Enterprise Edition を必要とせずに、信頼性の高いフェイルオーバーソリューションを提供できます。代わりに、FCI では SQL Server Standard Edition ライセンスのみが必要です。FCI を使用して、SQL Server のライセンスコストを 65～75% 削減できます。

> **Note**
>
> SQL Server エディション間のコストの違いの詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。

以下の点を考慮してください。

- Amazon FSx for Windows File Server は、SQL Server FCI 共有ストレージ要件を満たすための強力なソリューションを提供します。FSx for Windows File Server を使用すると、ストレージレプリケーションソリューションのライセンスを購入し、共有ストレージを自分で管理する必要がなくなります。これにより、30～40% の大幅なコスト削減につながります。詳細については、 AWS ストレージブログの「Amazon FSx for Windows File Server を使用した Microsoft SQL Server の高可用性デプロイの簡素化」の投稿を参照してください。
- 「Software Assurance benefits summary」(ダウンロード可能な PDF) と Bring Your Own License (BYOL) モデルを使用すると、セカンダリサーバーがパッシブである限り、パッシブフェイルオーバーのメリットを活用できます。これにより、クラスターのパッシブノードにライセンスを提供する必要がないため、SQL ライセンスのコスト削減につながります。

次の図は、FSx for Windows File Server を使用した SQL Server FCI のアーキテクチャの例を示しています。



SIOS DataKeeper

SQL Server FCIsデプロイする場合は、共有ストレージ要件を検討することをお勧めします AWS。従来のオンプレミスインストールでは通常、共有ストレージ要件を満たすためにストレージエリアネットワーク (SAN) が使用されますが、これは AWSでは実行可能なオプションではありません。Amazon FSx for Windows File Server は、 上の SQL Server FCI に推奨されるストレージソリューションですが AWS、異なる にクラスターサーバーを追加することを妨げる制限がありますAWS リージョン。

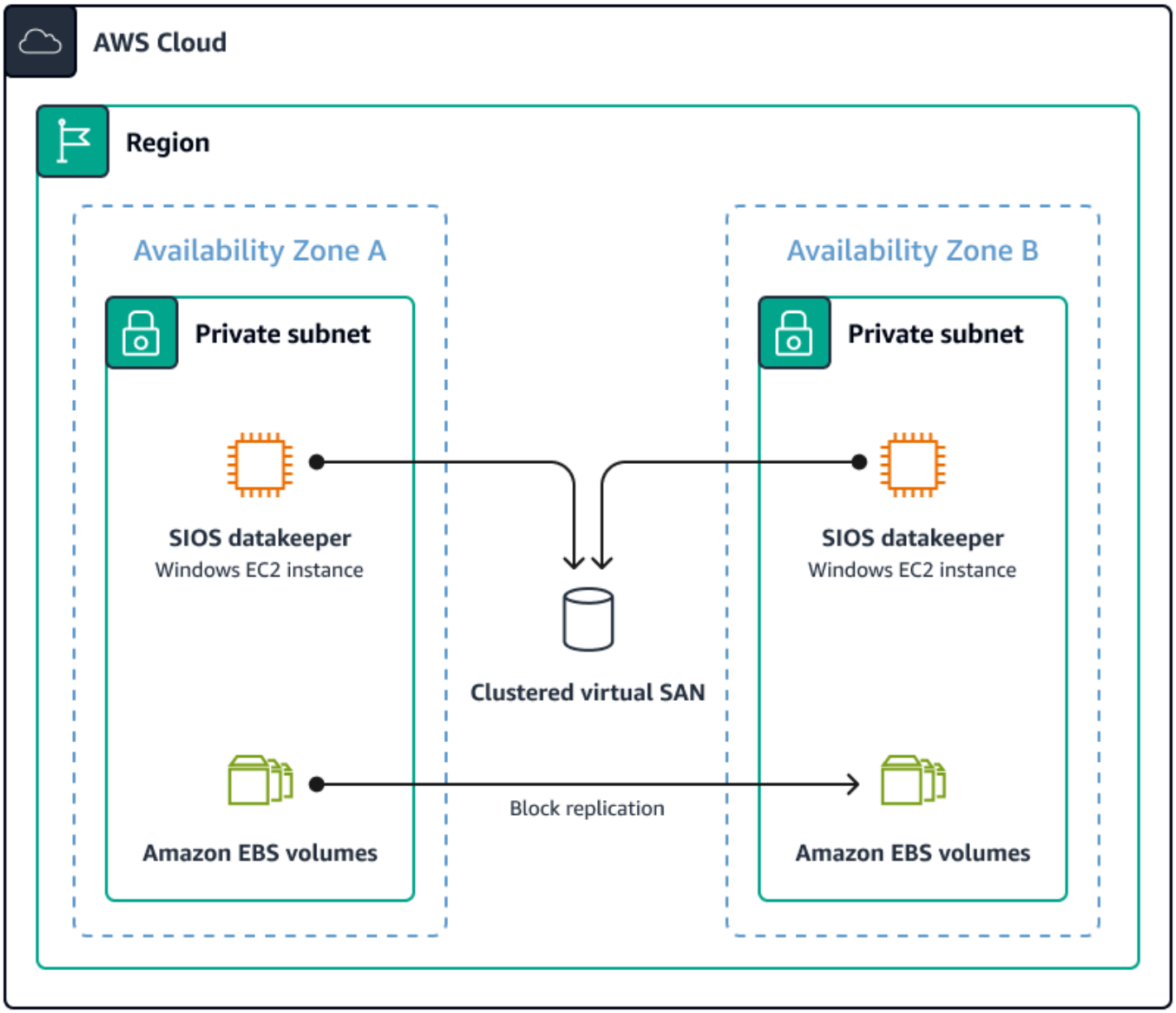
SIOS DataKeeper を使用して、コストを 58～71% 削減しながら、アベイラビリティーゾーンとリージョンの両方をカバーする SQL Server FCI を作成できます。SIOS DataKeeper は、FCI の高可用性のメリットを実現するのに役立ちます。これにより、SIOS DataKeeper は組織にとってコスト効率と信頼性に優れたソリューションになります。

SIOS DataKeeper を使用することで得られる次のような利点も考慮してください。

- SIOS DataKeeper は、ローカル EBS ボリュームを使用して、クラスター化された仮想 SAN を作成し、アベイラビリティーゾーン間の同期レプリケーションを使用して高可用性を実現します。ディザスタリカバリの場合、SIOS DataKeeper はリージョン間の非同期レプリケーションを使用します。
- SIOS DataKeeper は、SQL Server Standard Edition を使用してエンタープライズクラスのクラスタリング機能を提供します。これにより、SQL Server Enterprise Edition を使用する SQL Server Always On 可用性グループによる高可用性の実装と比較して、SQL Server のライセンスコストが 65～75% 削減されます。SIOS DataKeeper を使用すると、組織のニーズを満たす、可用性が高くて柔軟で費用対効果の高い SQL Server 環境を作成できます。

> **Note**
>
> SQL Server エディション間のコストの違いの詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。

次の図は、クラスター化された仮想 SAN ソリューションを使用した SQL Server FCI のアーキテクチャの例を示しています。

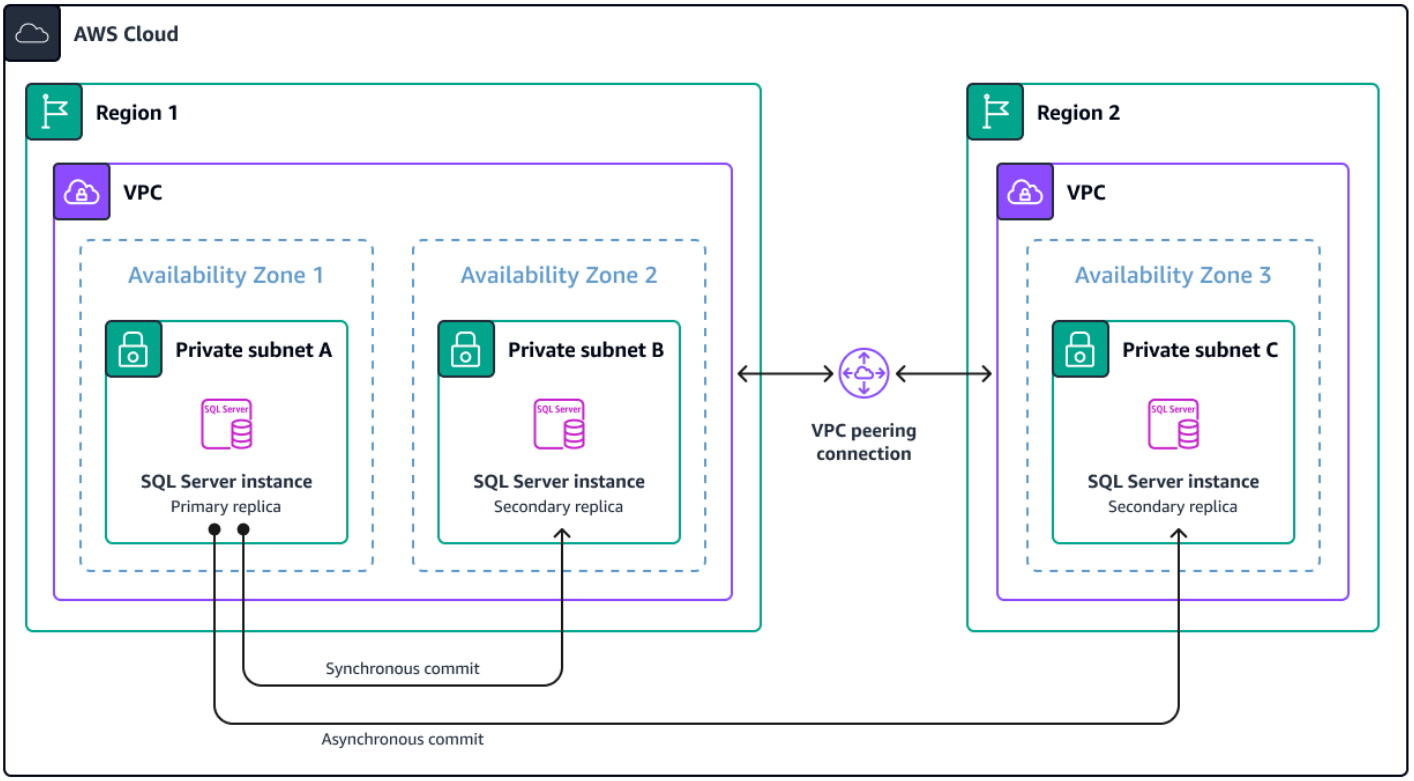


Always On 可用性グループ

Always On 可用性グループは、高可用性とディザスタリカバリの両方の目的で使用できます。SQL Server を 1 つのリージョンの 2 つのアベイラビリティーゾーンにデプロイすることで、高可用性を実現できます。リージョン間で可用性グループを拡張することで、ディザスタリカバリを実現できます。

次の図は、Always On 可用性グループに基づくソリューションのアーキテクチャの例を示しています。図のリージョン 1 のレプリカは、可用性グループの自動フェイルオーバーを提供する同期コ

ミットを使用しています。リージョン 2 のレプリカは非同期コミットを使用しているため、可用性グループの手動フェイルオーバーが必要になります。



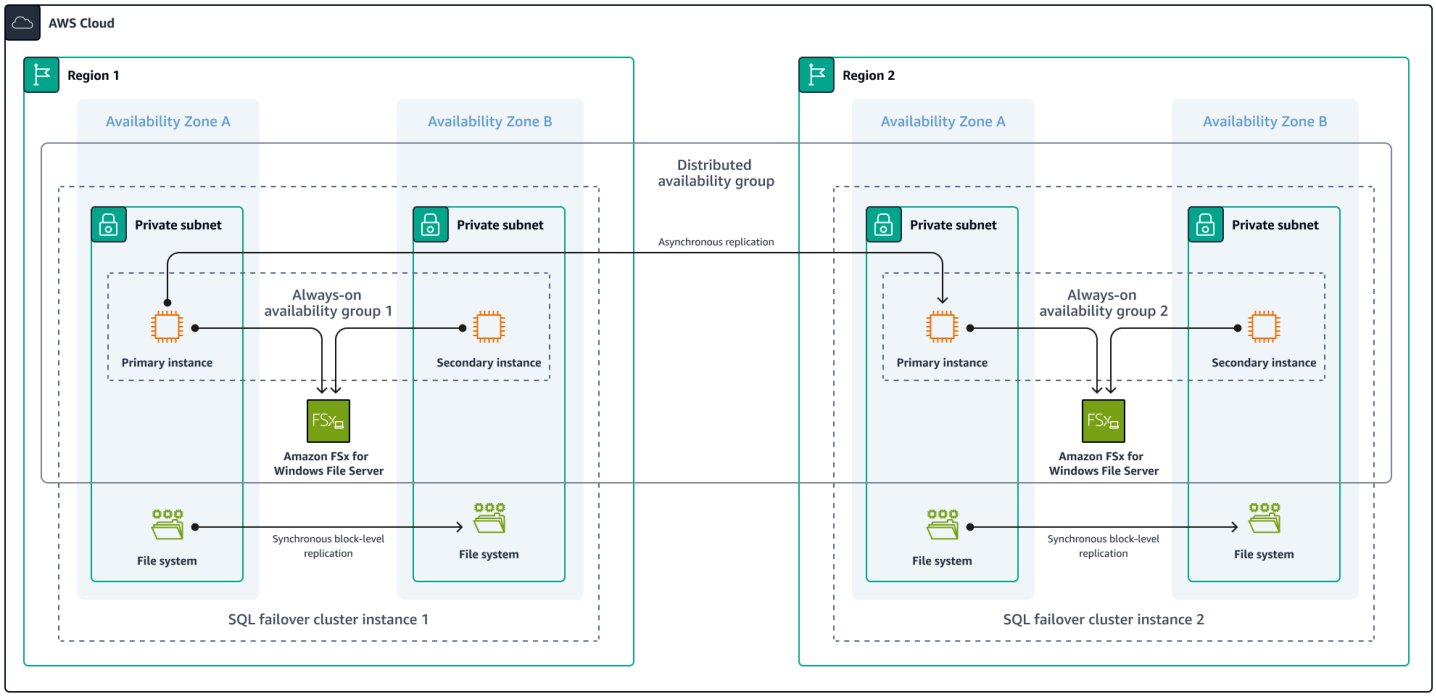
分散可用性グループ

信頼性やディザスタリカバリで妥協できないミッションクリティカルな SQL Server デプロイの場合は、マルチリージョンアプローチをお勧めします。複数のリージョンに可用性グループを分散させることは、ビジネス継続性を維持し、ダウンタイムを最小限に抑えるための最も回復力の高いソリューションです。

このアーキテクチャは、共有ストレージ、同期ブロックレベルレプリケーション、SQL Server FCI など、Amazon FSx for Windows File Server の機能を最大限に活用します。これらの機能により、複数のアベイラビリティーゾーンにまたがる高可用性 SQL Server 環境を作成できるようになります。この設定を別のリージョンにレプリケートすることで、最も深刻な中断にも対応できる完全に冗長なシステムが得られます。このソリューションを際立たせているのは、それが提供する柔軟性とセキュリティのレベルです。分散可用性グループのドメインに依存しないアーキテクチャにより、基盤となる Windows クラスターサーバーを異なる Active Directory ドメインに参加させることができると同時に、証明書ベースの認証により、SQL Server 環境に対する最大限の保護が保証され、マルチリージョン DR 戦略に高い RTO および RPO 要件が提供されます。マルチリージョンアーキテクチャの構築の詳細については、「 アーキテクチャブログ」の「Field Notes: Building a Multi-Region

Architecture for SQL Server using FCI and Distributed Availability Groups」を参照してください。AWS

次の図は、分散可用性グループを使用したマルチリージョンソリューションのアーキテクチャの例を示しています。



ログ配布

ログ配布は、予期しない停止が発生した場合にリージョン間でデータベースを保護するための、実績があり、信頼性が高く、費用対効果の高い方法です。組織は数十年にわたってログ配布を使用してデータを保護してきました。

ログ配送を実装する場合 AWS、トランザクションの頻度とログ配送ジョブに応じて、RPO と RTO を数分で実現できます。万一、リージョンにアクセスできなくなった場合、ログ配布によりデータの安全性と回復性が維持されます。

ログ配布を使用することで得られる次のような利点も考慮してください。

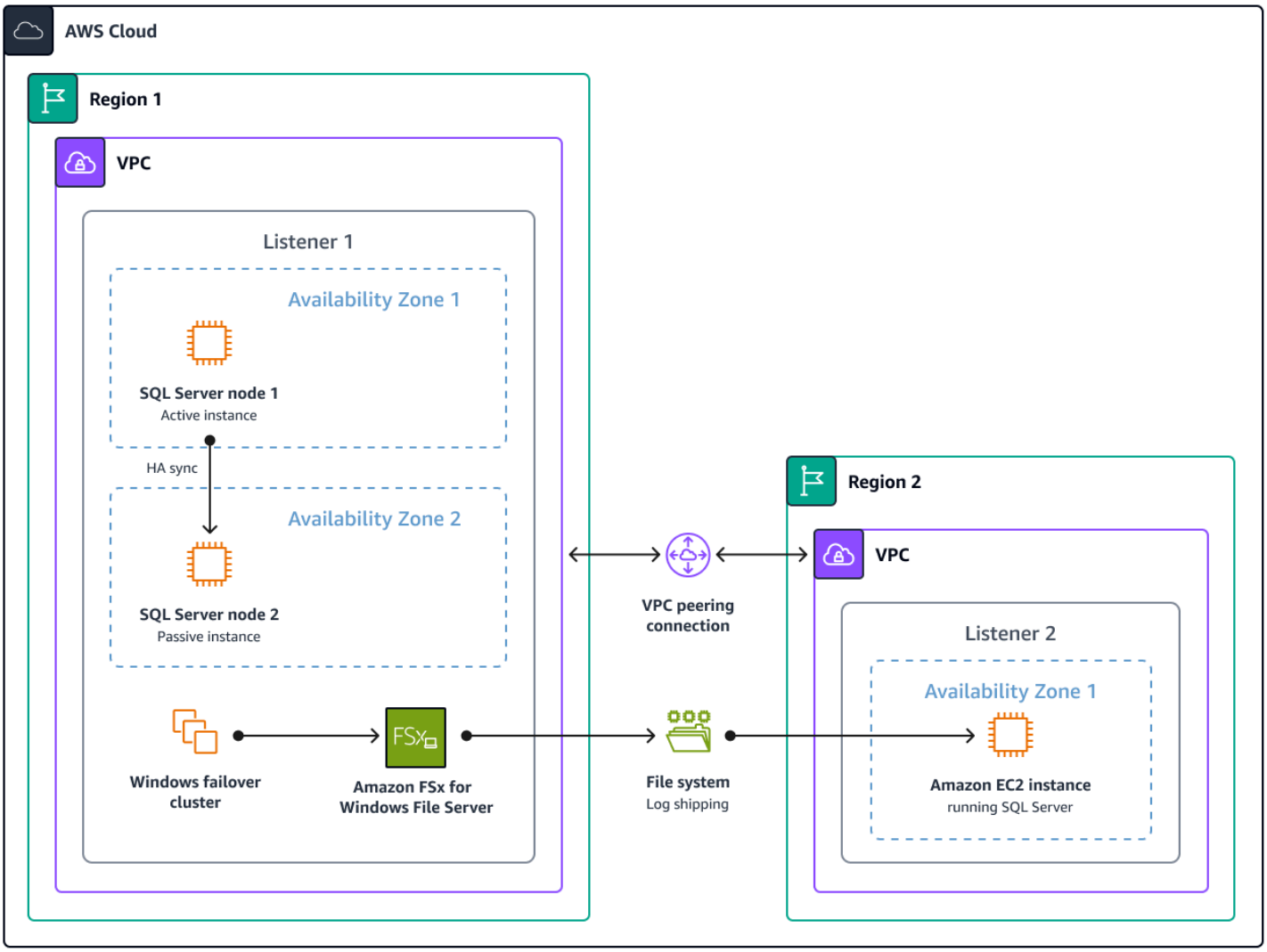
- リージョン間のディザスタリカバリ耐障害性のためにログ配布を使用することで、コストを削減し、ビジネス要件を満たします。SQL Server Standard Edition または SQL Server Web Edition のライセンスのみが必要なため、ログ配布は TCO を削減します。

- アクティブなソフトウェアアシュアランスでのログ配布を使用して、ディザスタリカバリ/パッシブサーバーからライセンスコストを削減します。ソフトウェアアシュアランスでのログ配布を使用する場合は、プライマリ/アクティブ SQL Server のみをライセンスする必要があります。
- SQL Server Enterprise Edition でリージョン間で分散可用性グループをセットアップする必要がなくなるため、SQL Server のライセンスコストが 65～75% 削減されます。これを行うには、SQL Server Standard Edition と SQL Server FCI をログ配布と組み合わせて使用して、ディザスタリカバリ要件を満たします。

> **Note**
>
> SQL Server エディション間のコストの違いの詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。

詳細については、 AWS アーキテクチャブログの「Amazon FSx for Windows 設定で SQL Server FCI のログ配信を使用して SQL Server DR を拡張する」を参照してください。

次の図は、ログ配布ソリューションのアーキテクチャの例を示しています。

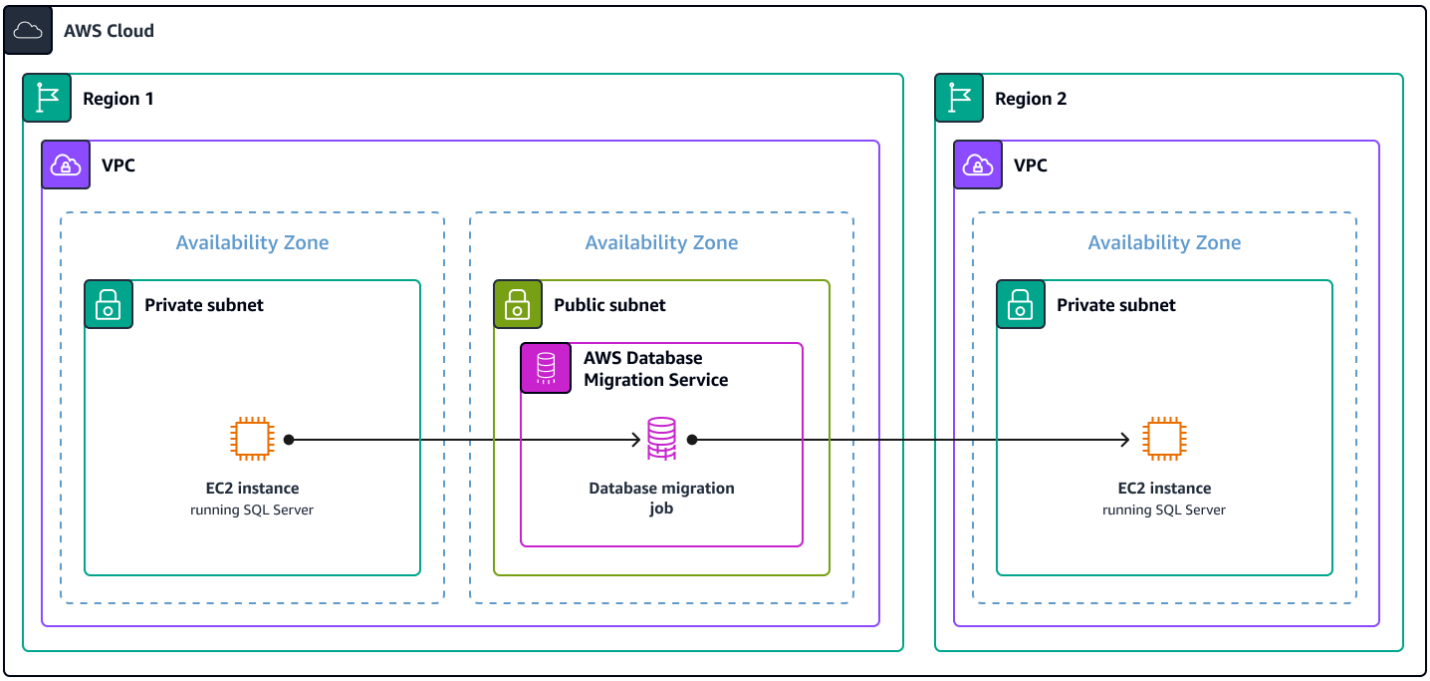


AWS Database Migration Service

AWS Database Migration Service (AWS DMS) を使用して、アプリケーションのニーズに基づいて HA/DR ソリューションを設計できます。 AWS DMS を使用すると、同じリージョン (HA) またはリージョン (DR) 間でセカンダリ SQL Server データベースにデータを簡単にコピーできます。このアプローチは技術的に健全であり、リソース使用量を最適化しながら AWS インフラストラクチャへの投資を最大化できます。

AWS DMS はコスト効率の高いサービスです。転送プロセス中に使用される CPU リソースと追加のログストレージに対してのみ課金されます。つまり、大幅な追加コストを発生させることなく、このソリューションのメリットを享受できます。を使用して AWS DMS 、ライセンスとリソースの使用に関連するコストを最小限に抑えながら、データの可用性とアクセス性を確保できます。

次の図は、 AWS DMSに基づくソリューションのアーキテクチャの例を示しています。



AWS Elastic Disaster Recovery

組織によっては、すべての重要なビジネスアプリケーションにディザスタリカバリ計画が導入されていることを確認する必要があります。以前は、このような組織の多くは、従来のディザスタリカバリソリューションに多額の投資を行っていました。この場合、重複するインフラストラクチャ全体を事前に構築して維持する必要があります。このアプローチはコストも時間もかかり、スケーリングも困難です。

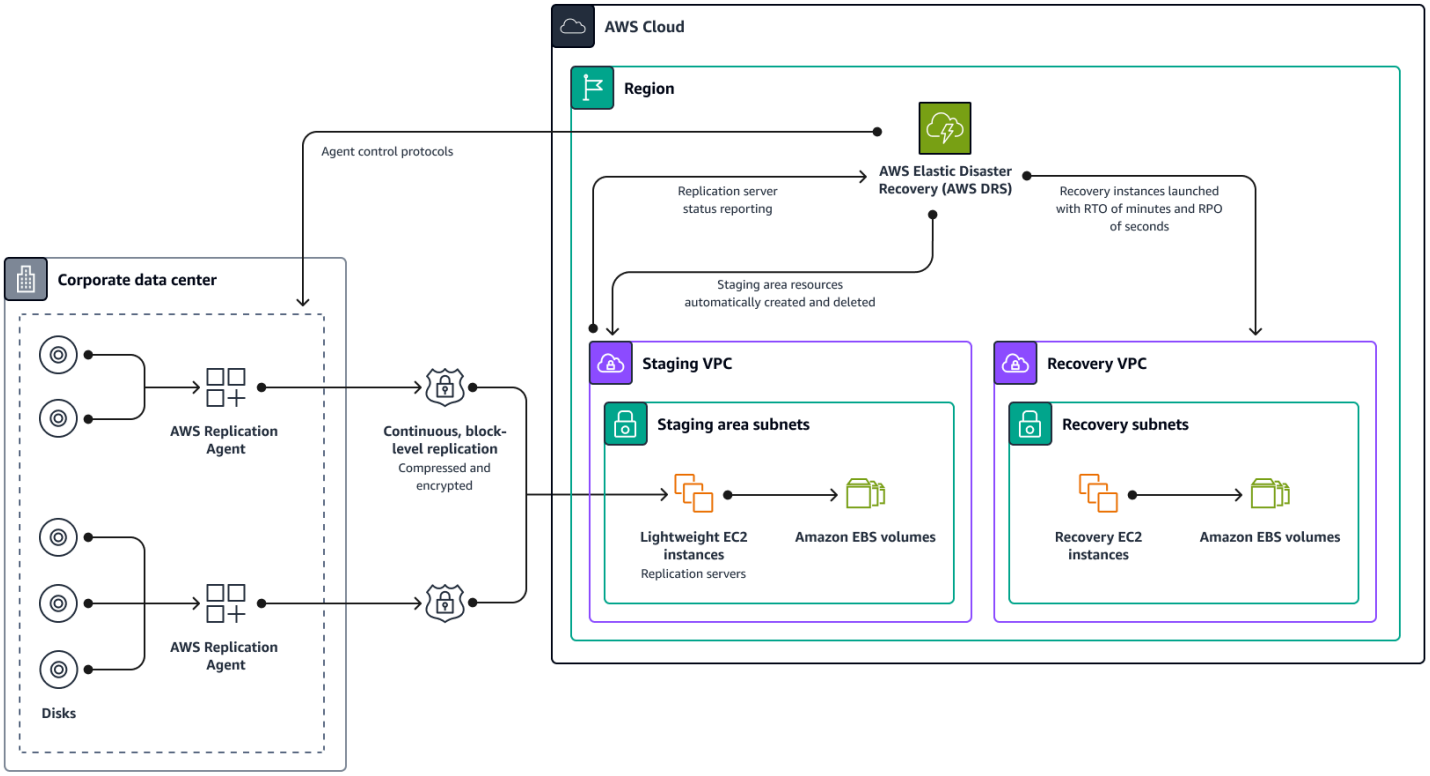
これで、 AWS Elastic Disaster Recovery を使用してディザスタリカバリインフラストラクチャを事前に構築する必要がなくなります。ディザスタリカバリマシンは、必要になるまで Elastic Disaster Recovery で起動されないため、必要なときに使用した分だけお支払いいただきます。つまり、ソフトウェアライセンスと高性能コンピューティングのコストを大幅に削減できます。

さらに、ディザスタリカバリソリューションのステージングエリアには、低コストの Amazon Elastic Block Store (Amazon EBS) ボリュームが含まれています。EBS ボリュームは、重複リソースのプロビジョニングコストをさらに削減します。これにより、ビジネス要件を満たす堅牢で信頼性の高いディザスタリカバリソリューションを維持しながら、全体的なディザスタリカバリコストを削減できます。Elastic Disaster Recovery を使用してコアビジネス活動に集中できますが、 AWS はディザスタリカバリソリューションの基盤となるインフラストラクチャを処理します。

SQL Server では、費用対効果の高いディザスタリカバリオプションとして Elastic Disaster Recovery を使用できます。耐障害性があり可用性の高い SQL Server アーキテクチャのパッシブノードのライセンスは、アクティブなソフトウェアアシュアランスを使用する場合にカバーされま

す。ただし、パッシブサーバーがオンラインになるためのコンピューティングコストは引き続き発生します。Elastic Disaster Recovery を使用すると、プライマリサーバーは、アクティブなソフトウェアアシュアランスを維持しなくても DR 環境にレプリケートでき、ディザスタリカバリのコンピューティングコストを支払う必要もありません。このコスト削減の組み合わせにより、SQL Server のディザスタリカバリコストを 50% 以上削減できます。

次の図は、Elastic Disaster Recovery に基づくソリューションのアーキテクチャの例を示しています。



詳細については、 AWS ブログの Microsoft Workloads の「 を使用して復元された DR サイトで SQL Server の高可用性を設定する方法 AWS Elastic Disaster Recovery」を参照してください。

コスト比較

次の表は、このセクションで説明する HA/DR ソリューションのコストを比較したものです。この比較では、次の前提が適用されます。

- インスタンスタイプ – r5d.xlarge
- ライセンスタイプ – Windows と SQL Server の両方に含まれるライセンス
- [Region] (リージョン) - `us-east-1`

ソリューション	高可用性	ディザスタリカバリ	Enterprise Edition	Standard Edition	Cost
ログ配布	いいえ	はい	はい	はい	SQL Server Enterprise Edition: 32,674.8 USD (2 ノード) SQL Server Standard Edition: 14,804.4 USD (2 ノード)
Always On 可用性グループ	はい	はい	はい	あり。ただし、基本的な可用性グループ (2 ノード)	SQL Server Enterprise Edition: 32,674.8 USD (2 ノード) SQL Server Standard Edition: 14,804.4 USD (2 ノード)
Always On FCI	はい	なし	はい	あり (2 ノード)	SQL Server Standard Edition: 14,804.4 USD

ソリューション	高可用性	ディザスタリカバリ	Enterprise Edition	Standard Edition	Cost
分散可用性グループ	はい	はい	あり	いいえ	SQL Server Enterprise Edition: 65,349.6 USD (4 ノード)
Elastic Disaster Recovery	いいえ	はい	はい	はい	1 インスタンスと 1 TB のストレージのレプリケーションの場合、月額約 107.48 USD 注: Elastic Disaster Recovery は、レプリケートするサーバーごとに時間単位で請求されます。ディスク数、ストレージサイズ、ドリルまたはリカバリの起動数、レプリケートするリージョンに関係なく、コストは同じです。

ソリューション	高可用性	ディザスタリカバリ	Enterprise Edition	Standard Edition	Cost
SIOS DataKeeper	はい	はい	はい	はい	ソフトウェアアシュアランスを使用する Always On 可用性グループ (2 ノード、24 コア): 213,480 USD SIOS DataKeeper とソフトウェアアシュアランスを使用する SQL Server Standard Edition で実行される 2 ノード SQL Server クラスター: 61,530 USD (2 ノード)
AWS DMS	いいえ	はい	はい	はい	r5.xlarge インスタンスと 1 TB のストレージの場合、月額 745.38 USD

コスト最適化の推奨事項

組織の要件を満たす HA/DR ソリューションを選択するには、次のステップを実行することをお勧めします。

- このガイドの「SQL Server ワークロードに適した EC2 インスタンスを選択する」セクションを確認します。
- ピークワークロード中にパフォーマンスカウンターを実行して、ワークロードの IOPS とスループットの要件を決定します。
  - IOPS = ディスク読み取り/秒 + ディスク書き込み/秒
  - スループット = ディスク読み取りバイト/秒 + ディスク書き込みバイト/秒
- パフォーマンスとコスト削減を向上させるには、次のストレージボリュームタイプを使用します。
  - `tempdb` およびバッファプール拡張用の NVMe インスタンスストレージ
  - データベースファイル用の io2 ボリューム
- Amazon EC2 上の SQL Server のコスト最適化に関する推奨事項については、AWS Trusted Advisor を使用します。SQL Server 最適化チェックを実行する Trusted Advisor ために、 のエージェントをインストールする必要はありません。 は、仮想 CPUs (vCPUs)、バージョン、エディションなど、Amazon EC2 SQL Server のライセンス込みインスタンス設定 Trusted Advisor を検査します。次に、 はベストプラクティスに基づいてレコメンデーション Trusted Advisor を行います。
- Amazon EC2 インスタンスと Amazon EBS の両方の適切なサイジングレコメンデーション AWS Compute Optimizer に使用します。
- AWS 料金見積りツール を使用して、コスト見積もりのための HA/DR 戦略を設計します。
- SQL Server Enterprise Edition から SQL Server Standard Edition へのダウングレードが可能なオプションかどうかを判断するには、sys dm_db_persisted_sku_features 動的管理ビューを使用して、現在のデータベースでアクティブなエディション固有の機能を特定します。

> **Note**
>
> ライセンス込み EC2 インスタンスを使用する場合、SQL Server エディションの変更にはサイドバイサイド移行が必要です。

- 定義された RTO と RPO でデータベースを復旧できる設計をより適切に構築するために、半年または年に 1 回のディザスタリカバリドリルを実行します。これは、アーキテクチャの弱点を特定するのにも役立ちます。

その他のリソース

- Amazon FSx for Windows File Server を使用して Microsoft SQL Server の高可用性デプロイを簡素化する (AWS ストレージブログ)
- フィールドノート: FCI と分散可用性グループを使用した SQL Server のマルチリージョンアーキテクチャの構築 (AWS アーキテクチャブログ)
- SQL Server のディザスタリカバリの設計 AWS: パート 1 (AWS データベースブログ)
- 「Microsoft SQL high availability with Amazon FSx for Windows」(YouTube)
- 「Maximizing Microsoft SQL Server Performance with Amazon EBS」(AWS ストレージブログ)
- オンプレミスストレージパターンと AWS ストレージサービスの比較 (AWS ストレージブログ)
- データセンター NAS を Amazon FSx File Gateway に置き換える計画 (AWS ストレージブログ)
- での高可用性 SQL Server デプロイのコストの最適化 AWS (AWS ストレージブログ)
- 「How to set up disaster recovery for SQL Server Always On Availability Groups using AWS Elastic Disaster Recovery」(Microsoft Workloads on AWS)
- を使用して復元された DR サイトで SQL Server の高可用性を設定する方法 AWS Elastic Disaster Recovery (Microsoft ワークロードオン AWS)

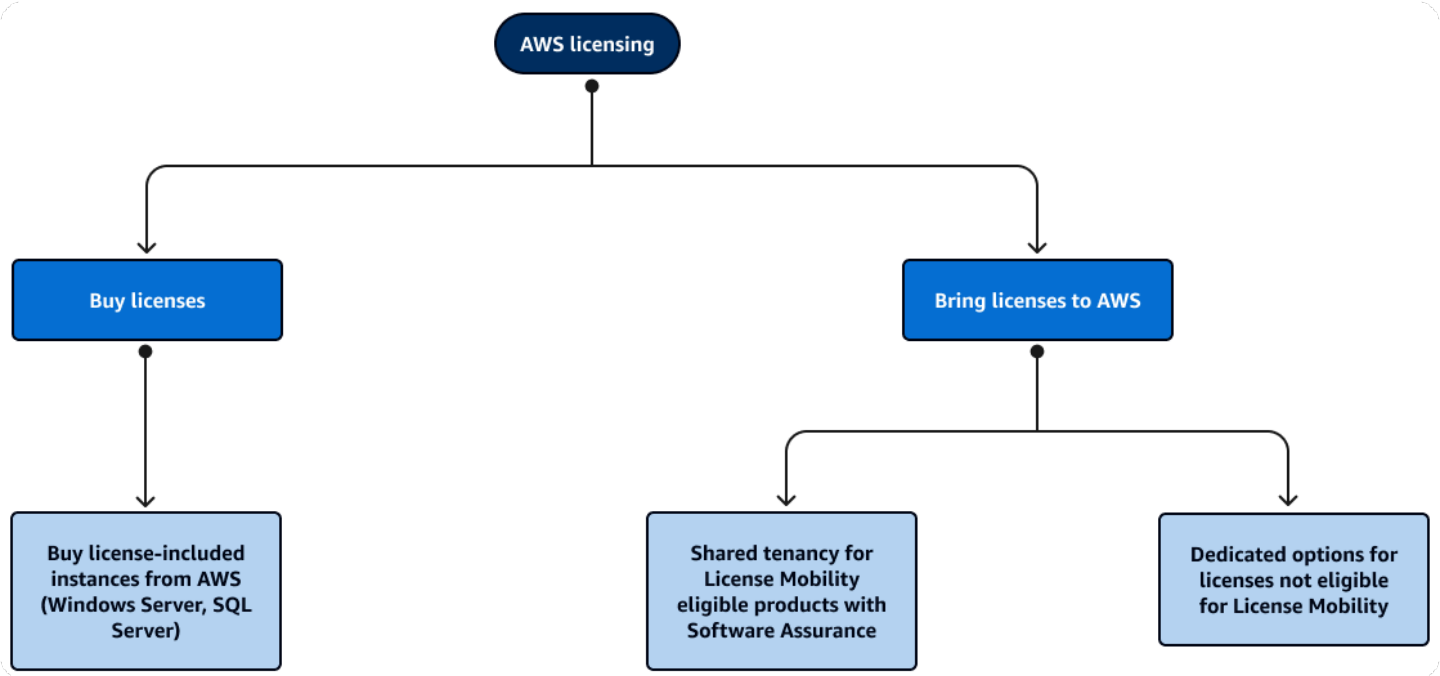
SQL Server ライセンスを理解する

概要

ワークロードをクラウドに移行する企業が増えるにつれて、クラウドプラットフォームのコストの最適化が最優先事項となっています。ライセンスは、Microsoft ワークロードの実行に関連する最も重要なコストの 1 つです AWS。このセクションでは、SQL Server の Microsoft ライセンスを最適化 AWS して のコストを最適化する方法について説明します。

AWS ライセンスオプション

AWS は、ライセンス用にさまざまな柔軟なコスト最適化の選択肢を提供します。これらのライセンスオプションは、コストを削減し、コンプライアンスを維持し、ビジネスニーズを満たすのに役立つように設計されています。



AWS はライセンスを 3 つの主なタイプに分類します。

1. ライセンス込み – このライセンスオプションを使用すると、ライセンスをオンデマンドで購入して使用でき、使用した分に対してのみ料金が発生します。ライセンス込みオプションは、ライセンスの使用に柔軟性が必要で、前払いコストを回避したいシナリオに最適です。Windows Server、SQL Server、その他の Microsoft 製品から選択できます。
2. ライセンスモビリティを備えた Bring Your Own License (BYOL) 製品 – このライセンスオプションは、既存のライセンスがすでにあり、クラウドで使用するシナリオ向けに設計されています。AWS では、Microsoft のライセンスモビリティプログラムを通じて独自のライセンスをクラウドに持ち込むことができます。SQL Server with Software Assurance (SA) などのライセンスモビリティを持つ製品を共有テナンシーまたは専用テナンシーに持ち込んで、 AWS インスタンスコストを削減できます。
3. ライセンスモビリティのない BYOL 製品 – Windows Server などのライセンスモビリティを持たない Microsoft 製品には、クラウドでこれらの製品を使用するための専用オプション AWS が用意されています。さらに、専有ホストは、物理コアレベルでライセンスする機会を提供します。これにより、ワークロードの実行に必要なライセンス料を 50% 以上節約できます。専有ホストは、ほぼ常時実行される安定した予測可能なワークロードに最適なオプションです。

ライセンス持ち込みによるコストへの影響

ライセンスの持ち込みは、 AWSで Microsoft ワークロードを実行するコストに大きな影響を与える可能性があります。独自のライセンスを持ち込む場合、クラウドで実行されているインスタンスに対して追加のライセンスコストを支払う必要はありません。これにより、大幅なコスト削減につながる可能性があります。

次の比較は、1 つの c5.xlarge インスタンスを 24 時間 365 日実行した場合のオンデマンドの月額コストを示しています。

- Windows Server + SQL Server Enterprise Edition: 1,353 USD/月 (ライセンス込み)
- Windows Server + SQL Server Standard Edition: 609 USD/月 (ライセンス込み)
- Windows Server のみ: 259 USD/月 (ライセンス込み)
- コンピューティングのみ (Linux): 127 USD/月

結局のところ、独自のライセンスを持ち込むと、 AWSで Microsoft ワークロードを実行するコストに大きな影響を与える可能性があります。既存のライセンスを使用すると、ライセンスコストを削減し、全体的な AWS 請求額を節約できます。

ライセンスの最適化

AWS 最適化とライセンス評価 (AWS OLA) は、コンピューティングとライセンスのコストを削減することで、ライセンスの最適化に役立ちます。 AWS OLA は、 で実行されているワークロード AWS 、または移行が予定されているワークロードのライセンス要件を評価するように設計されています。 AWS OLA は、ライセンスの使用を最適化するための推奨事項を提供します。

ライセンスの使用を最適化するための重要な戦略の 1 つは、インスタンスの適切なサイズ設定です。適切なサイズ設定を行うには、CPU、メモリ、およびストレージの要件に基づいて、ワークロードに適したインスタンスタイプを選択する必要があります。適切なインスタンスサイズを選択することで、リソースをコスト効率の高い方法で使用できるようになります。これにより、大幅なコスト削減につながる可能性があります。

Microsoft ソフトウェアライセンスでは、ソフトウェアが実行されるコアの数は、ライセンスコストを決定する上で重要な要素です。例えば、Windows Server および SQL Server ライセンスは通常、コア数に基づいてライセンスされます。インスタンスのサイズを適切に設定することで、Microsoft ソフトウェアが実行されるコアの数を減らし、インスタンスのコストと必要なライセンスの数の両方を減らすことができます。

コスト最適化の推奨事項

ライセンスの最適化は、 AWSでのコスト最適化の重要な要素です。適切な戦略を実装することで、ライセンスコストを削減し、コンプライアンスを維持し、ライセンス投資から最良の価値を実現することができます。このセクションでは、ライセンス最適化のいくつかの戦略の概要を説明します。

対象となる Windows Server ライセンスの持ち込み

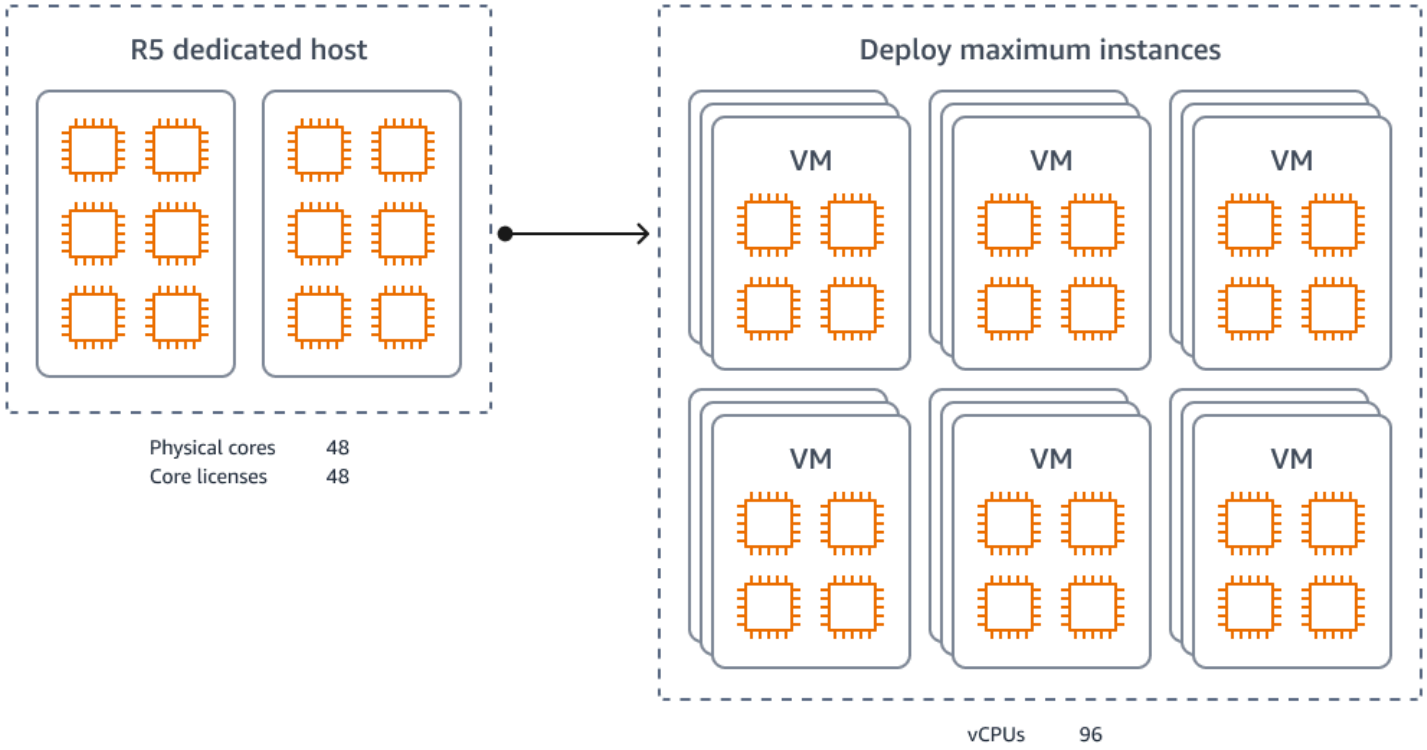
独自の Windows Server ライセンスの持ち込みは、ライセンス最適化の最も効果的な戦略の 1 つです。この戦略により、既存の投資を活用して AWS 支出を削減できます。

例えば、2019 年 1 月 10 日より前にライセンスを購入した場合、またはその日付より前に署名された有効な Enterprise 契約の下でライセンスを調整として購入した場合は、Windows Server 2019 以前のバージョンを Amazon EC2 専有ホストにデプロイできます。このルールは、Microsoft が 2019 年に、Windows Server などのライセンスモビリティのない製品のライセンス条件を、出品プロバイダー (Alibaba AWS、Google Cloud など) にデプロイした場合に行った変更に基づいています。新しい条件では、独自の Windows Server ライセンスを に持ち込むことはできません AWS が、代わりにライセンス込みインスタンスを使用する必要があります。ただし、その日より前に永続的ライセンスを購入した場合は、引き続きそれらの Windows Server ライセンスを Amazon EC2 専有ホストにデプロイできます。

物理レベルのライセンス

物理コアレベルでライセンスすることで、ホストの物理コアのみをライセンスできるため、必要なライセンス数に影響を与えることなく、最大数のインスタンスをデプロイできます。これは通常、Windows Server Datacenter および SQL Server Enterprise Edition を使用して行われます。

例として、48 個のコアを持つ R5 専有ホスト (96 個の vCPU に相当) について考えてみましょう。Windows Server Datacenter Edition を使用する場合、必要なライセンスは 48 個のみです。これにより、次の図に示すように、最大 96 個の vCPU を持つインスタンスの組み合わせをデプロイできるようになります。



このアプローチは、ホストで実行できるインスタンスの数を最大化するのに十分なワークロードがある場合に、特に費用対効果が高くなります。物理コアレベルでライセンスすることで、インスタンスごとの追加のライセンスコストを回避し、ライセンス投資に対して最良の価値を実現できます。

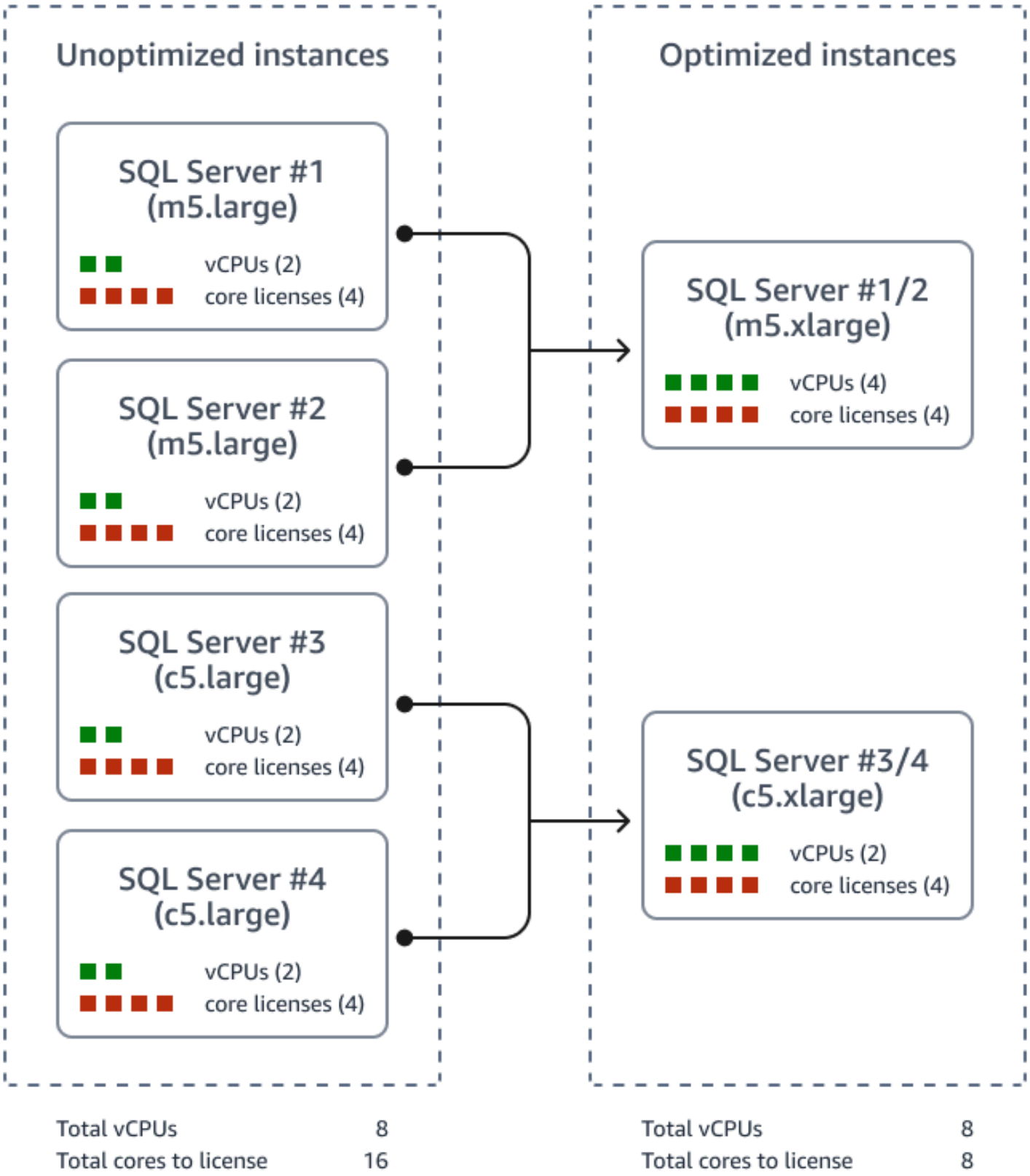
SQL Server の物理コアレベルでのライセンス

共有テナンシーでは、SQL Server ライセンスは、インスタンスに割り当てられた vCPU の数に基づきます。これに対し、専有ホストでは、物理コアレベルまたは vCPU レベルで SQL Server Enterprise Edition をライセンスできます。

前述の R5 専有ホストの例と同様に、物理コアレベルで SQL Server Enterprise Edition をライセンスする場合、ホストのライセンスに必要な SQL Server Enterprise Edition ライセンスは 48 個のみです。これに対し、共有テナンシー (オプションは vCPU 単位のライセンスのみ) では、同じワークロードに対して 96 個の SQL Server Enterprise Edition ライセンスが必要です。したがって、専有ホストは、共有テナンシーと比較して SQL Server のライセンスコストを最大 50% 削減できます。これは、対象となる Windows ライセンスを持ち込むことでインスタンスコストを節約することに加えて行われます。

SQL Server インスタンスの統合

SQL Server の統合は、複数の SQL Server インスタンスを 1 つのサーバーに結合するプロセスです。SQL Server では、インスタンスに vCPU が 2 つしかない場合でも、インスタンスごとに最低 4 つのコアライセンスが必要です。つまり、4 コア未満のサーバーで SQL Server を実行すると、これらのインスタンスを過剰にライセンスし、必要以上のライセンスを使用することになります。


Unoptimized instances
SQL Server #1 (m5.large)
vCPUs (2)
core licenses (4)
SQL Server #2 (m5.large)
vCPUs (2)
core licenses (4)
SQL Server #3 (c5.large)
vCPUs (2)
core licenses (4)
SQL Server #4 (c5.large)
vCPUs (2)
core licenses (4)
Optimized instances
SQL Server #1/2 (m5.xlarge)
vCPUs (4)
core licenses (4)
SQL Server #3/4 (c5.xlarge)
vCPUs (2)
core licenses (4)
Total vCPUs 8
Total cores to license 16
Total vCPUs 8
Total cores to license 8

例えば、それぞれ 2 つの vCPU を持つ 2 つのインスタンスを、4 つの vCPU を持つ 1 つのインスタンスに統合すると、ライセンス要件を 50% 削減できます。これは、8 つではなく 4 つのコアライセンスのみ必要となるためです。

統合の詳細については、このガイドの「SQL Server consolidation」セクションを参照してください。

SQL Server エディションのダウングレード

SQL Server エディションの変更は、ライセンスの使用を最適化し、コストを削減するための重要な戦略となります。SQL Server の Enterprise エディションは Standard エディションよりもかなり高価であるため、ダウングレードすると大幅なコスト削減につながる可能性があります。

透過的なデータ暗号化 (TDE) と Always On 可用性グループは、SQL Server Enterprise Edition で人気の高い 2 つの機能です。ただし、SQL Server Enterprise Edition の完全な機能セットが必要ない場合は、これらの機能に代わるコスト効率の高い代替方法を検討することもできます。例えば、TDE は、SQL Server 2019 以降の SQL Server Standard Edition で利用できます。Always On 可用性グループの代わりに、FSx for Windows File Server の共有ストレージでフェイルオーバークラスタリングを使用することで、SQL Server Standard Edition で高可用性を実現できます。

SQL Server Enterprise Edition から SQL Server Standard Edition にダウングレードすることで、ライセンスコストを大幅に削減できます。詳細については、 AWS ストレージブログの記事の「高可用性 SQL Server デプロイのコストの最適化 AWS」を参照してください。

ライセンスコストの削減に加えて、SQL Server エディションをダウングレードすると、ソフトウェアアシュアランスの支出を削減し、将来の調整を回避することができます。未使用のライセンスをシェルフに返せば、追加のライセンスコストを回避し、ライセンス投資から最良の価値を得ることができます。

SQL Server ワークロードを慎重に評価し、ビジネスニーズにとって重要な機能を判断することが重要です。詳細については、 AWS 「 規範ガイダンス」の「環境の評価」を参照してください。また、Microsoft SQL Server データベースが SQL Server Enterprise エディション固有の機能を使用しているかどうかを判断します。

SQL Server の適切なエディションを選択し、SQL Server Enterprise Edition の機能に代わるものを使用すると、コンプライアンスを維持し、ビジネスニーズを満たすと同時に、大幅なコスト削減を実現できます。ダウングレードオプションの詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。

非本番環境での SQL Server Developer Edition の使用

非本番環境では、オンプレミス環境で MSDN サブスクリプションを使用して、Enterprise Edition や Standard Edition などの SQL Server のライセンス可能エディションをデプロイできます。ただし、MSDN サブスクリプションにはライセンスモビリティはありません。したがって、 に移行する場合 AWS、これらのライセンスを引き継ぐことはできません。代わりに SQL Server Developer Edition を使用する必要があります。

SQL Server Developer Edition は、SQL Server のフル機能のエディションで、無料で利用できます。このエディションは、SQL Server バージョン 2016 以降で使用できます。また、Microsoft のウェブサイトからダウンロードできます。SQL Server Developer Edition は、本番環境のライブデータに接続していない限り、開発、テスト、ステージングなど、すべての非本番環境で使用することを目的としています。

SQL Server Developer Edition を非本番環境で使用した場合、追加のライセンスコストを回避できます。詳細については、このガイドの「SQL Server Developer Edition を評価する」セクションを参照してください。

SQL Server ワークロードの CPU の最適化

RAM やネットワークの制限などの他の要因により、ワークロードに必要な数よりも多くの CPU を持つインスタンスタイプを選択しなければならない場合があります。ただし、 AWS は、このような状況でライセンスコストを最適化するのに役立つソリューションを提供します。

SQL Server コアライセンスを持ち込むほとんどのお客様と同様に、EC2 インスタンスでハイパースレッディングを無効にするか CPU をオフにして、ホストで使用できる CPU の数を制限できます。このオプションを使用すると、追加のライセンスを購入するコストを節約しながら、RAM などの他のインスタンス機能を利用できます。

たとえば、ワークロードに必要なメモリが 128 GB で、SQL Server のコアが 8 つしかないために r5.4xlarge インスタンスをデプロイする場合、アクティブな CPUs が 8 つしかないインスタンスのハイパースレッディングを無効にすることができます。これにより、アクティブに使用されている 8 個のコアのライセンスのみが必要となるため、必要な SQL Server ライセンスを 50% 削減できます。

インスタンスタイプ	vCPU の合計	CPU の最適化機能を備えたアクティブな vCPU	SQL Server ライセンスの節約
r5.4xlarge	16	8	50%
r5.12xlarge	48	8	83%

CPU の最適化機能は、Amazon EC2 起動設定中、または既存のインスタンスを変更することで設定できます。また、BYOL インスタンスとライセンス込みの Amazon EC2 インスタンスの両方に適用することもできます。この柔軟性により、CPU をワークロードのニーズに合わせると同時に、Windows Server および SQL Serverライセンスを削減できます。ライセンス込みの Amazon EC2 インスタンスの場合、CPUsことでライセンスコストを即座に削減できます。

インスタンスのサイズを適正化すると、ワークロードに対して最も費用対効果の高いインスタンスタイプを確実に使用できるようになります。に新しいインスタンスタイプ AWS が導入されたため、これらの新しいインスタンスがより少ないコアでワークロード要件を満たすことができるかどうかを評価することが重要です。

その他のリソース

- アマゾン ウェブ サービスと Microsoft: よくある質問 (AWS ドキュメント)

SQL Server ワークロードに適した EC2 インスタンスを選択する

> **Important**
>
> このセクションを読む前に、このガイドの「SQL Server ライセンスを理解する」セクションと「Select the right instance type for Windows workloads」セクションを読んでおくことをお勧めします。

概要

Microsoft SQL Server は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで 15 年以上実行されています。 AWS はその経験を積んでおり、最小限の仕様から高パフォーマンスのマルチ

リージョンクラスターまで実行される SQL Server ワークロードに合わせて Amazon EC2 インスタンスを開発するのに役立ちます。

SQL Server の正しい EC2 インスタンスの選択は、ワークロードに大きく依存します。SQL Server のライセンス方法、メモリの使用方法、そして SQL Server の機能が Amazon EC2 サービスとどのように連携しているかを理解することは、アプリケーションに最適な EC2 インスタンスを導くのに役立ちます。

このセクションでは、さまざまな SQL Server ワークロードと、ライセンスとコンピューティングのコストを最小限に抑えるために特定の EC2 インスタンスと組み合わせる方法について説明します。

コスト比較

Amazon EC2 では、Bring Your Own License (BYOL) を使用するか、Windows Server および SQL Server ライセンスで従量制料金を使用することができます。従量制料金ライセンスの場合、Windows Server および SQL Server ライセンスのライセンスコストは、EC2 インスタンスの時間単位のコストに組み込まれます。例えば、異なる料金の異なる AMI を持つことができます。AMI の料金は、AMI が実行される SQL Server エディションによって異なります。

Windows Server と SQL Server の料金は項目化されません。AWS 料金見積りツール のようなツールには項目別料金はありません。ライセンス込みサービスのさまざまな組み合わせを選択した場合は、次の表に示すようにライセンスコストを推定できます。

EC2 インスタンス	AMI	コンピューティング料金	Windows ライセンス料金	SQL ライセンス料金	合計料金
r5.xlarge	Linux (コンピューティング料金)	183.96 USD	-	-	183.96 USD
r5.xlarge	Linux + SQL Developer	183.96 USD	$0	$0	183.96 USD
r5.xlarge	Windows Server (LI)	183.96 USD	134.32 USD	-	318.28 USD
r5.xlarge	Windows + SQL Developer	183.96 USD	134.32 USD	$0	318.28 USD

EC2 インスタンス	AMI	コンピューティング料金	Windows ライセンス料金	SQL ライセンス料金	合計料金
r5.xlarge	Windows + SQL Web (LI)	183.96 USD	134.32 USD	49.64 USD	367.92 USD
r5.xlarge	Windows + SQL Standard (LI)	183.96 USD	134.32 USD	350.4 USD	668.68 USD
r5.xlarge	Windows + SQL Enterprise (LI)	183.96 USD	134.32 USD	1,095 USD	1413.28 USD

> **Note**
>
> 前の表の料金は、us-east-1 リージョンのオンデマンド料金に基づいています。

SQL Server を実行する最も費用対効果の高い方法は、上位エディションの機能が必要になるまで、下位エディションのままにしておくことです。詳細については、このガイドの「SQL Server エディションの比較」セクションを参照してください。SQL Server Web Edition から SQL Server Standard Edition へのアップグレードは、SQL Server ライセンスコストの 7 倍を超え、Standard Edition から Enterprise Edition への移行コストの 3 倍を超えます。ライセンスコストの格差は考慮すべき重要な要素であり、このセクションの残りの部分で説明します。

コスト最適化シナリオ

配送車両を追跡する分析会社が SQL Server のパフォーマンスの向上を検討しているというシナリオの例を考えてみましょう。MACO の専門家が会社のパフォーマンスのボトルネックを確認した後、会社は x1e.2xlarge インスタンスから x2iedn.xlarge インスタンスに移行します。インスタンスサイズは小さくなりますが、x2 インスタンスの機能強化により、バッファプール拡張機能を使用することで SQL Server のパフォーマンスと最適化が向上します。これにより、同社は、SQL Server Enterprise Edition から SQL Server Standard Edition にダウングレードし、SQL Server ライセンスを 8 vCPU から 4 vCPU に削減することができました。

最適化前:

サーバー	EC2 インスタンス	SQL Server エディション	月額コスト
ProdDB1	x1e.2xlarge	Enterprise	3,918.64 USD
ProdDB2	x1e.2xlarge	Enterprise	3,918.64 USD
合計			7,837.28 USD

最適化後:

サーバー	EC2 インスタンス	SQL Server エディション	月額コスト
ProdDB1	x2iedn.xlarge	標準	1,215.00 USD
ProdDB2	x2iedn.xlarge	標準	1,215.00 USD
合計			2,430.00 USD

x1e.2xlarge インスタンスから x2iedn.xlarge インスタンスへの変更を組み合わせることで、サンプル顧客は本番データベースサーバーで 1 か月あたり 5,407 USD を節約できました。これにより、ワークロードの総コストが 69% 削減されました。

> **Note**
>
> 前の表の料金は、`us-east-1` リージョンのオンデマンド料金に基づいています。

コスト最適化の推奨事項

メモリ最適化インスタンス

SQL Server の最も重要な側面の 1 つは、メモリへの依存を理解することです。SQL Server は、オペレーティングシステムで使用されていない使用可能なすべての RAM を使用しようとします (デフォルトのインストールでは最大 2 TB)。これはパフォーマンス上の理由から行われます。メモリ内

のデータを操作する方が、絶えずディスクからデータをプルし、変更を加えてディスクに書き戻すよりもはるかにパフォーマンスが高くなります。代わりに、SQL Server はアタッチされたデータベースからできるだけ多くのデータをロードしようとし、そのデータを RAM に保持します。データに対する変更はメモリ内で実行され、後でディスクに書き込まれます。

> **Note**
>
> SQL Server が変更を書き込む方法の詳細については、Microsoft ドキュメントの「Writing Pages」を参照してください。

SQL Server は RAM 容量が大きいほどパフォーマンスが向上するため、通常は Amazon EC2 メモリ最適化インスタンスタイプから始めることをお勧めします。メモリ最適化インスタンスは汎用性が高く、さまざまなオプションを提供します。R ファミリーは、vCPU と RAM の比率が 1 対 8 で、インテルプロセッサ、AMD プロセッサ、拡張ネットワーキング、拡張 EBS パフォーマンス、インスタンスストレージ、拡張プロセッサ速度のオプションがあります。メモリ負荷の高いワークロード向けに、同じオプションを多数組み合わせ、vCPU と RAM の比率を 1 対 32 に拡張する X ファミリーもあります。メモリ最適化インスタンスは汎用性があるため、あらゆる形状およびサイズの SQL Server ワークロードに適用できます。

最小リソース未満 (4 vCPU 未満) のワークロード

一部のユースケースはバースト可能 (T3) インスタンスでうまく機能しますが、通常は SQL Server ワークロードにバースト可能インスタンスを使用しないことをお勧めします。SQL Server のライセンスは、インスタンスに割り当てられた vCPU の数に基づいています。SQL Server が 1 日の大半がアイドル状態で、バーストクレジットを取得している場合は、完全に活用されていない SQL ライセンスに対して料金が発生します。さらに、SQL Server には、サーバーあたり 4 コアの最小ライセンス要件があります。つまり、4 vCPU 分のコンピューティング能力を必要としない SQL Server ワークロードがある場合、使用していない SQL Server ライセンスを支払っていることになります。これらのシナリオでは、より大きなサーバーに複数の SQL Server インスタンスを統合するのが最適です。

最小リソース (64 GB 未満の RAM) を使用するワークロード

64 GB RAM 未満の SQL Server ワークロードの多くは、高パフォーマンスまたは高可用性を優先しません。これらのタイプのワークロードでは、アプリケーションが Microsoft のライセンス制限の対象である場合、SQL Server Web Edition が適している可能性があります。

> **Important**
>
> SQL Server Web Edition には、Microsoft のライセンス条項に基づいてユースケースが制限されています。SQL Server Web Edition のライセンスは、パブリックアクセスやインターネットアクセスが可能なウェブページ、ウェブサイト、ウェブアプリケーション、およびウェブサービスをサポートします。基幹業務アプリケーション (顧客関係管理、企業リソース管理、その他の類似アプリケーションなど) のサポートには使用できない可能性があります。

SQL Server Web Edition は、最大 32 vCPU および 64 GB RAM までスケール可能で、SQL Server Standard Edition よりも 86% 安価です。リソースの少ないワークロードでは、Intel の同等のものと比較してコンピューティング料金が 10% 安価な r6a などの AMD メモリ最適化インスタンスを使用することも、コンピューティングと SQL のライセンスコストを最小限に抑える良い方法です。

平均リソース (128 GB 未満の RAM) のワークロード

SQL Server Standard Edition は、最大 128 GB の RAM の SQL Server ワークロードの大部分で使用されます。SQL Server Standard Edition は、SQL Server Enterprise Edition よりも 65～75% 安価で、最大 48 vCPU および 128 GB RAM までスケールできます。通常、128 GB RAM の制限は 48 vCPU の制限より前に発生するため、この点が、SQL Server Enterprise Edition へのアップグレードを避けたいほとんどのお客様の焦点となります。

SQL Server には、バッファプール拡張機能と呼ばれる機能があります。この機能により、SQL Server はディスクの一部を使用して RAM の拡張として機能させることができます。バッファプール拡張機能は、Amazon EC2 インスタンスストレージで使用される NVMe SSD などの超高速ストレージと組み合わせると、うまく機能します。インスタンスストレージを含む Amazon EC2 インスタンスは、インスタンス名に「d」と表示されます (r5d、r6id、x2iedn など)。

バッファプール拡張機能は、通常の RAM に代わるものではありません。ただし、128 GB を超える RAM が必要な場合は、r6id.4xlarge や x2iedn.xlarge などの EC2 インスタンスでバッファプール拡張機能を使用して、Enterprise Edition ライセンスへのアップグレードを遅らせることができます。

高パフォーマンスワークロード (128 GB を超える RAM)

高いパフォーマンスを必要とする SQL Server ワークロードは、多くのリソースに依存するため、コスト最適化が困難です。ただし、EC2 インスタンスの違いを理解することで、間違った選択を防ぐことができます。

次の表は、さまざまなメモリ最適化 EC2 インスタンスとそのパフォーマンス制限を示しています。

	r5b	r6idn	r7iz	x2iedn	x2iezn
プロセッサ	3.1 GHz 第 2 世代インテル Xeon プロセッサ	3.5 GHz 第 3 世代インテル Xeon プロセッサ	3.9 GHz 第 4 世代インテル Xeon スケーラブルプロセッサ	3.5 GHz 第 3 世代インテル Xeon プロセッサ	4.5 GHz 第 2 世代インテル Xeon プロセッサ
CPU と RAM の比率	1:8	1:8	1:8	1:32	1:32
最大 vCPU	96	128	128	128	48
最大 RAM	768 GB	1,024 GB	1,024 GB	4,096 GB	1,536 GB
インスタンスストレージ	–	NVMe SSD (4x 1900 GB)	–	NVMe SSD (2x 1900 GB)	–
io2 Block Express	サポート対象	サポート対象	サポート対象	サポート	–
最大 EBS IOPS	260,000	350,000	160,000	260,000	80,000
最大 EBS スループット	60 Gbps	80 Gbps	40 Gbps	80 Gbps	19 Gbps
最大ネットワーク帯域幅	25 Gbps	200 Gbps	50 Gbps	100 Gbps	100 Gbps

各インスタンスは異なる目的に使用されます。SQL Server ワークロードを理解すると、最適なインスタンスタイプを選択するのに役立ちます。

属性の詳細:

- r5b – r5b の「b」属性は、このインスタンスタイプが高 EBS パフォーマンスに焦点を当てていることを意味します。第 5 世代のメモリ最適化インスタンスでは、r5b が優先選択肢でした。こ

れは、io2 Block Express ボリュームを使用し、最大ストレージ IOPS が 260,000 に達した最初のインスタンスタイプでした。r5b インスタンスタイプは、依然として、高 EBS パフォーマンスのニーズに対するコスト効率の高い代替手段です。

- r6idn – 第 6 世代のメモリ最適化インスタンスは、前世代よりも大幅に改善されました。r5b からの EBS パフォーマンスの強化は、r6idn でさらに一歩前進し、最大 IOPS が 350,000 に向上しました。r6idn には、SQL Server のパフォーマンスをさらに向上させるために、tempdb およびバッファプール拡張用のインスタンスストアボリュームもあります。
- x2iedn – x2iedn は r6idn に似ています。同様のレベルの拡張 EBS、拡張ネットワーキング、NVMe SSD インスタンスストレージを提供しますが、高メモリワークロードと低 CPU 量 (SQL Server ライセンスコストが低い) では vCPU と RAM の比率が 1:32 になります。
- x2iezn – x2iezn の「z」属性は、このインスタンスタイプが高プロセッサパフォーマンスに焦点を当てていることを示します。Cascade Lake プロセッサの全コアターボ周波数は最大 4.5 GHz です。vCPU の数を低く抑えたいシナリオでは、この EC2 インスタンスを 1:32 の vCPU と RAM の比率と組み合わせて使用することをお勧めします。これにより、SQL Server のライセンスコストを低く抑えることができます。
- r7iz – r7iz の「z」属性は、このインスタンスタイプが高プロセッサパフォーマンスに焦点を当てていることを示します。Sapphire Rapids プロセッサの全コアターボ周波数は最大 3.9 GHz です。x2iezn インスタンスと同様に、r7iz は高周波プロセッサのパフォーマンスを優先しますが、vCPU と RAM の比率は 1:8 です。

その他のリソース

- 汎用 Amazon EC2 インスタンス (AWS ドキュメント)
- 「Comparison tool」(Vantage)
- ライセンス – SQL Server (AWS ドキュメント)

インスタンスを統合する

このセクションでは、複数の SQL Server インスタンスを同じサーバーに結合してライセンスコストを最小限に抑え、リソース使用率を最大化するコスト最適化手法に焦点を当てます。

概要

インスタンスの作成は、SQL Server データベースエンジンをインストールするプロセスの一部です。SQL Server インスタンスは、独自のサーバーファイル、セキュリティログイン、システムデー

タベース (マスター、モデル、msdb、tempdb) を含む完全なインストールです。インスタンスには独自のファイルとサービスがすべて備わっているため、インスタンスの相互の干渉なく、同じオペレーティングシステムに複数の SQL Server インスタンスをインストールできます。ただし、インスタンスはすべて同じサーバーにインストールされるため、コンピューティング、メモリ、ネットワークなどの同じハードウェアリソースを共有します。

通常、本番環境ではサーバーごとに 1 つの SQL Server インスタンスのみを使用し、「ビジー」インスタンスが共有ハードウェアリソースを過剰に使用しないようにします。各 SQL Server インスタンスに、独自のリソースを持つ独自のオペレーティングシステムを提供することは、リソースガバナンスに依存するよりも優れた境界となります。これは、大量の RAM と CPU のリソースを必要とする高パフォーマンス SQL Server ワークロードに特に当てはまります。

ただし、すべての SQL Server ワークロードが大量のリソースを使用するわけではありません。例えば、一部の組織では、コンプライアンスまたはセキュリティの目的で、各顧客に独自の専用 SQL Server インスタンスを割り当てています。小規模なクライアントや通常はアクティブではないクライアントの場合は、最小限のリソースで SQL Server インスタンスを実行することになります。

「Microsoft SQL Server 2019: Licensing guide」に記載されているように、SQL Server を実行している各サーバーは、最低 4 つの CPU ライセンスを考慮する必要があります。つまり、2 つの vCPU のみでサーバーを実行する場合でも、4 つの vCPU の SQL Server ライセンスを取得する必要があります。Microsoft が公開している SQL Server の価格に基づくと、SQL Server Standard Edition を使用する場合の差額は 3,945 USD になります。最小限のリソースを使用して 1 つの SQL Server インスタンスで複数のサーバーを実行している組織では、未使用のリソースをライセンスしなければならない場合の合計コストが高額になる可能性があります。

コスト最適化シナリオ

このセクションでは、それぞれが 1 つの SQL Server インスタンスを持つ 4 つの Windows Server サーバーの実行と、複数の SQL Server インスタンスを同時に実行する 1 つの大きな Windows Server サーバーの実行の違いを比較するシナリオの例について説明します。

各 SQL Server インスタンスが 2 つの vCPU と 8 GB RAM のみを必要とする場合、サーバーあたりの合計コストは、SQL Server ライセンスに対して 7,890 USD、さらに 1 時間あたりのコンピューティング コストとして 0.096 USD になります。

EC2 インスタンス	vCPU	RAM	料金	ライセンスする vCPU	SQL Server ライセンスコストの合計
m6i.large	2	8	0.096	4	7,890 USD

これを 4 つのサーバーに拡張すると、合計コストは、SQL Server ライセンスに対して 31,560 USD、さらに 1 時間あたりのコンピューティング コストとして 0.384 USD になります。

EC2 インスタンス	vCPU	RAM	料金	ライセンスする vCPU	SQL Server ライセンスコストの合計
4x m6i.large	2	32	0.384	16	31,560 USD

4 つの SQL Server インスタンスをすべて 1 つの EC2 インスタンスに結合した場合、コンピューティングリソースとコンピューティングの合計量は同じままになります。ただし、不要な SQL Server ライセンスコストを排除することで、ワークロードを実行する合計コストを 15,780 USD 削減できます。

EC2 インスタンス	vCPU	RAM	料金	ライセンスする vCPU	SQL Server ライセンスコストの合計
m6i.2xlarge	8	32	0.384	8	15,780 USD

> **Note**
>
> 上記の表では、コンピューティングコストは、`us-east-1` リージョンで Windows Server を実行している Amazon EC2 サーバーの時間単位のオンデマンド料金を示しています。SQL Server Standard Edition のライセンスコストは、Microsoft が公開している SQL Server の価格を参照しています。

コスト最適化の推奨事項

SQL Server インスタンスの統合を検討している場合、最大の懸念事項は、統合する各インスタンスのリソース消費量です。各サーバーのワークロードパターンをよりよく理解するには、長期間にわたってパフォーマンスメトリクスを取得することが重要です。リソース消費モニタリングの一般的なツールには、Amazon CloudWatch、Windows Performance Monitor (perfmon)、SQL Server のネイティブモニタリングツールがあります。

SQL Server ワークロードを組み合わせて互いに干渉することなく同じサーバーリソースを使用できるかどうかを分析するときは、次の質問を検討することをお勧めします。

- 定常状態で消費されるリソースは何か (CPU、メモリ、ネットワーク帯域幅)。
- スパイク時に消費されるリソースは何か (CPU、メモリ、ネットワーク帯域幅)。
- スパイクはどのくらいの頻度で発生するか。スパイクは一貫しているか。
- あるサーバーのリソーススパイクが別のサーバーのリソーススパイクと一致しているか。
- SQL Server で使用されるストレージの IOPS とスループットはどれくらいか。

SQL Server インスタンスを組み合わせる計画を進める場合は、 AWS クラウド運用および移行ブログの記事「Run multiple instances of SQL Server on one Amazon EC2 instance」を参照してください。この記事では、SQL Server で設定を変更してインスタンスを追加する方法について説明しています。開始する前に、同じサーバーに複数のインスタンスがインストールされる場合の小さな違いを考慮してください。

- デフォルトの SQL Server データベースインスタンスの名前は `MSSQLSERVER` で、ポート 1433 を使用します。
- 同じサーバーにインストールされる各追加インスタンスは、「名前付き」データベースインスタンスです。
- 各名前付きインスタンスには、一意のインスタンス名と一意のポートがあります。
- 名前付きインスタンスへのトラフィックを調整するには、SQL Server Browser を実行する必要があります。
- 各インスタンスは、データベースデータファイル用に個別の場所を使用でき、個別のログインを使用できます。
- SQL Server の最大サーバーメモリ設定は、各インスタンスのパフォーマンスニーズに応じて設定する必要があり、合わせた合計において、基盤となるオペレーティングシステムに十分なメモリも残しておく必要があります。

- 移行や統合には、SQL Server のネイティブバックアップおよび復元機能、または AWS DMS を使用できます。

その他のリソース

- SQL Server Licensing Datasheet (AWS クラウドオペレーションと移行ブログ)
- SQL Server マルチインスタンスセットアップブログ記事 (AWS クラウドオペレーションと移行ブログ)

SQL Server エディションの比較

概要

Microsoft SQL Server ライセンスは、Windows ワークロード環境の最大のコストの 1 つです。SQL Server のライセンスコストは、ワークロードを実行するためのコンピューティングコストを簡単に超える可能性があります。間違ったエディションを選択すると、使用していない機能や不要な機能に対して料金が発生する可能性があります。このセクションでは、次の SQL Server エディションについて、機能や相対コストなどを比較します。

- Enterprise – SQL Server Enterprise Edition は、高パフォーマンス、無制限の仮想化、および複数のビジネスインテリジェンス (BI) ツールを備えたデータセンター機能を提供します。
- Standard – SQL Server Standard Edition は、小規模な組織や部門に基本的なデータ管理機能およびビジネスインテリジェンスを提供します。
- Web – SQL Server Web Edition は、ウェブホスティング業者またはウェブ付加価値プロバイダー (VAP) である企業に適しています。このエディションは、総保有コストが低く、小規模から大規模のウェブプロパティにスケーラビリティおよび管理機能を提供します。

> **Important**
>
> SQL Server Web Edition を使用すると、パブリックアクセスやインターネットアクセスが可能なウェブページ、ウェブサイト、ウェブアプリケーション、およびウェブサービスのみをサポートできます。SQL Server Web Edition を使用して、基幹業務アプリケーション (顧客関係管理アプリケーションやエンタープライズリソース管理アプリケーションなど) をサポートすることはできません。

- Developer – SQL Server Developer Edition には Enterprise Edition のすべての機能が含まれていますが、開発のみを目的としています。
- Express – SQL Server Express Edition は無料のデータベースであり、学習やデスクトップアプリケーションの構築に使用できます。Express Edition は、他のエディションにアップデートできます。

> **Note**
>
> SQL Server Evaluation Edition は、180 日間のトライアル期間で利用できます。

コストへの影響

Microsoft 販売代理店から SQL Server ライセンスを購入し、ソフトウェアアシュアランスを使用して AWS に持ち込むことができます。または、ライセンス込みの Amazon EC2 AMI が含まれる従量制料金モデルで SQL Server ライセンスを使用することもできます。

Microsoft 販売代理店から SQL Server ライセンスを購入する場合、コアライセンスは 2 つのパックで販売され、サーバーごとに最低 4 つのコアをライセンスする必要があります。次の表は、Enterprise Edition と Standard Edition のコストの比較を示しています。

バージョン	SQL Server Enterprise Edition (2 コアパック)	SQL Server Standard Edition (2 コアパック)	削減量
2022	15,123 USD	3,945 USD	74%
2019	13,748 USD	3,586 USD	74%

> **Note**
>
> 上記の表の料金は、Microsoft が公開している、SQL Server 2022 および SQL Server 2019 の価格に基づいています。

次のコスト比較は、ライセンス込みの Amazon EC2 AMI を使用して SQL Server のさまざまなエディションをホストする場合を示しています。この比較では、SQL Server は `us-east-1` リージョンの r6i.xlarge (4 vCPU) でホストされます。

インスタンス	コンピューティングコスト	Windows ライセンスコスト	SQL Server ライセンスコスト	Total
R6i.xlarge (Linux)	183.96 USD	–	–	183.96 USD
R6i.xlarge + Windows	183.96 USD	134.32 USD	–	318.28 USD
R6i.xlarge + SQL Server Web Edition	183.96 USD	134.32 USD	49.35 USD	367.63 USD
R6i.xlarge + SQL Server Standard Edition	183.96 USD	134.32 USD	350.4 USD	668.68 USD
R6i.xlarge + SQL Enterprise Edition	183.96 USD	134.32 USD	1,095 USD	1,413.28 USD

ワークロードに適した SQL Server エディションを選択することで、SQL Server のライセンスコストを最大 95% 削減できます。次の表は、r6i.xlarge インスタンスの SQL Server ライセンスのコストを比較したものです。

Edition	削減 (%)
Standard と Enterprise の比較	68%
Web と Standard の比較	86%
Web と Enterprise の比較	95%

ほとんどのシナリオでは、組織は Enterprise Edition から Standard Edition に切り替えますが、Standard Edition または Enterprise Edition から Web Edition への切り替えが可能な場合もあります。

コスト最適化の推奨事項

スケーリング制限、高可用性、パフォーマンス、セキュリティに基づいて、ワークロードに最適なエディションを選択できます。次の表は、SQL Server エディションでサポートされている機能を示しています。これは、使用するエディションを決定するのに役立ちます。この比較は、SQL Server 2016 SP1 以降のバージョンに適用されます。

[Scaling limits] (スケーリング履歴)

次の表は、さまざまな SQL Server エディションのスケーリング制限を比較したものです。

機能	Enterprise Edition	Standard Edition	Web Edition	Express Edition
SQL Server データベースエンジン、SQL Server Analysis Services (SSAS)、または SQL Server Reporting Services (SSRS) の単一のインスタンスで使用される最大コンピューティングキャパシティ	オペレーティングシステムの最大数	4 ソケットまたは 24 コアのいずれか小さい方に制限	4 ソケットまたは 16 コアのいずれか小さい方に制限	4 ソケットまたは 4 コアのいずれか小さい方に制限
SQL Server データベースエンジンのインスタンスあたりのバッ	オペレーティングシステムの最大数	128 GB	64 GB	1410 MB

機能	Enterprise Edition	Standard Edition	Web Edition	Express Edition
ファプールの最大メモリ				
SQL Server データベースエンジンのインスタンスあたりのバッファプール拡張機能の最大キャパシティ	最大メモリ設定の 32 倍	最大メモリ設定の 4 倍	該当なし	該当なし
最大リレーショナルデータベースサイズ	524 PB	524 PB	524 PB	10 GB
列ストアキャッシュまたはメモリ最適化データの最大メモリ	オペレーティングシステムの最大数	32 GB	16 GB	352 MB

アプリケーションが 16 コア (32 vCPU) 未満のコア数と 64 GB の RAM を必要とする場合は、SQL Server Web Edition から評価を開始できます。ワークロードが 64 GB を超えるメモリやその他の高可用性オプションを必要とする場合は、SQL Server Standard Edition にアップグレードする必要があります。

SQL Server Web Edition を使用して、パブリックアクセスやインターネットアクセスが可能なウェブページ、ウェブサイト、ウェブアプリケーション、ウェブサービスをサポートすることはできますが、SQL Server Web Edition を使用して基幹業務アプリケーションをサポートすることはできません。SQL Server Web Edition のユースケースの詳細については、Microsoft ライセンスサポートまたは Microsoft 販売代理店にお問い合わせください。

SQL Server Standard Edition は、最大 24 コア (48 vCPU) と 128 GB のメモリのワークロードに使用できます。ただし、バッファプール拡張機能を使用して、SQL Server Standard Edition が、r6id EC2 インスタンスに存在するようなローカルインスタンスストレージを利用できるようにすることができます。これにより、メモリは最大メモリ設定の 4 倍のサイズまで拡張されます。この機能の

組み合わせにより、メモリ要件が増加し始めたときにサーバーを Enterprise Edition にアップグレードするのを先延ばしにすることができます。

メモリ使用率を特定するには、バッファプールのデータベースページとページの平均寿命のカウンターを見つけます。ページの平均寿命は、ページがディスクにフラッシュされるまでにメモリ内にとどまる時間を示します。このカウンターのデフォルト値は 300 です。ページが数時間または数日間メモリに存在する場合、割り当てられたメモリを減らす可能性があります。

高可用性

次の表は、さまざまな SQL Server エディションの高可用性機能を比較したものです。

機能	Enterprise Edition	Standard Edition	Web Edition	Express Edition
サーバーコアサポート 1	はい	はい	はい	はい
ログ配布	はい	はい	あり	いいえ
データベースのミラーリング	はい	完全安全モード	ウィットネスとしてのみ	ウィットネスとしてのみ
バックアップ圧縮	はい	あり	なし	いいえ
Always On フェイルオーバークラスターインスタンス	16 ノード	2 ノード	いいえ	いいえ
Always On 可用性グループ	2 つの同期セカンダリレプリカを含む、最大 8 つのセカンダリレプリカ	いいえ	なし	いいえ
基本的な可用性グループ	いいえ	2 ノード	いいえ	いいえ

機能	Enterprise Edition	Standard Edition	Web Edition	Express Edition
オンラインページとファイルの復元	はい	なし	なし	いいえ
オンラインインデックス作成	はい	なし	なし	いいえ
オンラインスキーマの変更	はい	なし	なし	いいえ
高速リカバリ	はい	なし	なし	いいえ
ミラーリングされたバックアップ	はい	なし	なし	いいえ
メモリと CPU のホットアド	はい	なし	なし	いいえ
暗号化されたバックアップ	はい	あり	なし	いいえ
Microsoft Azure へのハイブリッドバックアップ (URL へのバックアップ)	はい	あり	なし	いいえ
ディザスタリカバリ用のフェイルオーバーサーバー	はい	あり	なし	いいえ
高可用性のためのフェイルオーバーサーバー	はい	あり	なし	いいえ

その他の一般的な機能

次の表は、さまざまな SQL Server エディションの最も一般的な機能を比較したものです。機能の広範なリストについては、Microsoft ドキュメントの「Editions and supported features of SQL Server 2019」を参照してください。

機能	Enterprise Edition	Standard Edition	Web Edition	Express Edition
(パフォーマンス) リソースガバナー	はい	なし	なし	いいえ
(セキュリティ) 透過的データベース暗号化 (TDE)	はい	あり	なし	いいえ
(セキュリティ) 拡張キー管理 (EKM)	はい	なし	なし	いいえ
(レプリケーション) Oracle の公開	はい	なし	なし	いいえ
(レプリケーション) ピアツーピアトランザクションレプリケーション	はい	なし	なし	いいえ
変更データキャプチャ	はい	あり	なし	いいえ

SQL Server Developer Edition

開発、QA、テスト、ステージング、UAT 環境など、非本番ワークロードはすべて、SQL Server Developer Edition を使用して SQL Server のライセンスコストを 100% 節約できます。SQL Server をダウンロードしたら、共有テナンシーを使用して EC2 インスタンスに SQL Server Developer Edition をインストールできます。SQL Server Developer Edition には、専用のインフラストラクチャは不要です。詳細については、このガイドの SQL Server Developer Edition に関する推奨事項を参照してください。

エディションの切り替え

既存のワークロードの場合、あるエディションから別のエディションに切り替えるには広範なテストが必要です。Enterprise Edition または Standard Edition で実行されているワークロードをチェックして、エディション固有の機能が使用されているか、それらの機能に代替ソリューションがあるかを確認するのがベストプラクティスです。例えば、データベースが Enterprise レベルの機能を使用しているかどうかを確認する場合は、次のコマンド例に示すように、すべてのデータベースで動的管理ビュー (DMV) を実行できます。

```
SELECT feature_name FROM sys.dm_db_persisted_sku_features; GO
```

SQL メンテナンスジョブの一環としてのオンラインインデックス再作成など、T-SQL でキャプチャできない Enterprise Edition の機能がいくつかあります。これらは手動で検証する必要があります。

移行に関する考慮事項

SQL Server をライセンスする方法によって、エディションを切り替えるためのオプションが決まります。SQL Server AMI を含む AMI では、EC2 インスタンスの料金にライセンスコストが含まれています。ライセンスコストは AMI にバインドされます。AWS 請求コードを使用して、AMI に含まれる SQL Server のバージョンを検証できます。 AWS ライセンス込みインスタンスの場合、オペレーティングシステム内で SQL Server エディションを変更しても、AMI に関連する請求は変更されません。SQL Server の新しいエディションを実行する AMI を使用して、データベースを新しい EC2 インスタンスに移行する必要があります。

独自のライセンスを持ち込む場合は、柔軟性が高まります。それでも通常は、新しいバージョンを実行している別の EC2 インスタンスに移行することをお勧めします。これにより、何かが計画どおりに進まない場合に簡単にフェイルバックできます。ただし、既存のサーバーを使用する必要がある場合は、SQL Server のサイドバイサイドインストールを実行し、インスタンス間でデータベースを移行することができます。サイドバイサイドエディションダウングレードの詳細な手順については、MSSQLTips ウェブサイトの「Edition Upgrade and Downgrade in SQL Server」を参照してください。

その他のリソース

- 「Editions and supported features of SQL Server 2022」(Microsoft Learn)
- 「sys.dm_db_persisted_sku_features (Transact-SQL)」(Microsoft Learn)
- 「Which Version of SQL Server Should You Use?」 (Brent Ozar Unlimited)
- AWS 料金見積りツール (AWS)

SQL Server Developer Edition を評価する

概要

SQL Server Developer Edition は、Enterprise Edition のすべての機能を含む SQL Server の無料エディションであり、非本番環境で使用できます。Microsoft Developer Network (MSDN) ライセンスを使用できないクラウドでは、SQL Server Developer Edition は、開発およびテストワークロードのライセンスを提供することなくコストを削減する優れた方法です。これは、大規模な開発環境とテスト環境を実行し、不要なコストを削減しようとするチームにとって特に当てはまります。

本番環境は、アプリケーションのエンドユーザー (インターネットウェブサイトなど) がアクセスする環境として定義され、そのアプリケーションのフィードバックの収集や受け入れテスト以外にも使用されます。本番環境を構成するその他のシナリオは次のとおりです。

- 本番データベースに接続する環境
- 本番環境のディザスタリカバリまたはバックアップをサポートする環境
- アクティビティのピーク時に本番環境にローテーションされるサーバーなど、少なくともある程度の時間、本番環境に使用される環境

ライセンスの詳細については、 AWS ドキュメントの「AWS と Microsoft に関するよくある質問」を参照してください。

コストへの影響

非本番ワークロードに SQL Server Developer Edition を使用すると、開発環境とテスト環境の現在の SQL Server ライセンスコストを 100% 節約できます。

SQL Server version	SQL Server Enterprise Edition (2 コアパック)	SQL Server Standard Edition (2 コアパック)	SQL Server Developer Edition
2022	15,123 USD	3,945 USD	空き
2019	13,748 USD	3,586 USD	空き

> **Note**
>
> 上記の表の料金は、Microsoft が公開している、SQL Server 2022 および SQL Server 2019 の価格に基づいています。

次の表は、`us-east-2` リージョンで 4 つの vCPU で実行され、オンデマンド料金を使用するさまざまな SQL Server エディションのコストを比較したものです。これは、ライセンス込みインスタンスに依存するシナリオに適用されます AWS。

EC2 インスタンス	AMI	コンピューティング料金	Windows ライセンス料金	SQL Server ライセンス料金	合計料金
r5.xlarge	Linux (コンピューティング料金)	183.96 USD	–	–	183.96 USD
r5.xlarge	Linux + SQL Server Developer Edition	183.96 USD	$0	$0	183.96 USD
r5.xlarge	Windows Server (LI)	183.96 USD	134.32 USD	–	318.28 USD
r5.xlarge	Windows + SQL Server	183.96 USD	134.32 USD	$0	318.28 USD

EC2 インスタンス	AMI	コンピューティング料金	Windows ライセンス料金	SQL Server ライセンス料金	合計料金
	Developer Edition				
r5.xlarge	Windows + SQL Server Web Edition (LI)	183.96 USD	134.32 USD	49.64 USD	367.92 USD
r5.xlarge	Windows + SQL Server Standard Edition (LI)	183.96 USD	134.32 USD	350.4 USD	668.68 USD
r5.xlarge	Windows + SQL Server Enterprise Edition (LI)	183.96 USD	134.32 USD	1,095 USD	1413.28 USD

コスト最適化シナリオ

データ整合性企業が新たな買収を行った後、新しく取得したワークロードをマネージドホスティングプロバイダーの現在の場所から移行し、 AWS クラウドの他のワークロードと統合したいと考えました。初期料金では、会社の SQL Server ワークロードは、現在のマネージドサービスプロバイダー AWS よりも で実行されるコストが 60% 高いことがわかりました。MACO SME は見積もりを評価し、顧客が実際に開発環境とテスト環境のマネージドホスティングプロバイダーの SQL Server ライセンスに対して料金を支払っていることがわかりました。この会社は、移行中に非本番ワークロードを SQL Server Developer Edition に切り替えることで、SQL Server のライセンスを 40% 削減しました。

Amazon EC2 に含まれる SQL Server ライセンス

ライセンス込み AMI を使用する EC2 インスタンスに SQL Server がある場合、Enterprise Edition から Developer Edition に直接変換することはできません。ライセンス込みインスタンスのライセンス

コストは AMI にバインドされます。SQL Server がオペレーティングシステム内からアンインストールされても、EC2 インスタンスには引き続きライセンスコストが請求されます。

Developer Edition に変換するには、SQL Server Developer Edition をダウンロードし、新しい EC2 インスタンスにインストールしてから、データベースを移行する必要があります。さまざまな方法を使用して EC2 インスタンス間で SQL Server データベースを移行できます。詳細については、「Microsoft SQL Server データベースの AWS クラウドへの移行」ガイドの「SQL Server データベースの移行方法」を参照してください。また、自動化 SQL Server Developer ソリューションを使用して、移行する新しいインスタンスを準備することもできます。

Amazon EC2 上の SQL Server BYOL

BYOL を使用する SQL Server インスタンスがある場合は、次のインプレース変換またはサイドバイサイドダウングレードオプションから選択できます。

- Microsoft ウェブサイトから SQL Server Developer Edition をダウンロードします。手動または自動インストールの手順については、 AWS ブログの記事「Automating SQL Server Developer deployments」を参照してください。
- SQL Server のネイティブバックアップと復元を使用して、データベースを移行するか、ある SQL インスタンスから別の SQL インスタンスにデータベースをデタッチ/アタッチします。
- 一括デプロイには自動化ツールを使用します。

> **Note**
>
> SQL Server Developer Edition は、非本番環境専用です。

その他のリソース

- EC2 に SQL Server Developer Edition をデプロイするための SQL Server Developer デプロイの自動化 (AWS ブログ)
- 「SQL 2022 pricing」(Microsoft)
- 「SQL 2019 pricing」(Microsoft)
- 「Licensing options」(Amazon EC2 での SQL Server)
- AWS 料金見積りツール (Amazon EC2 での SQL Server ドキュメント)
- 「Microsoft SQL Server 2019 Licensing guide」(Microsoft からダウンロード)

- SQL Server 2022 Developer Edition (Microsoft からダウンロード)

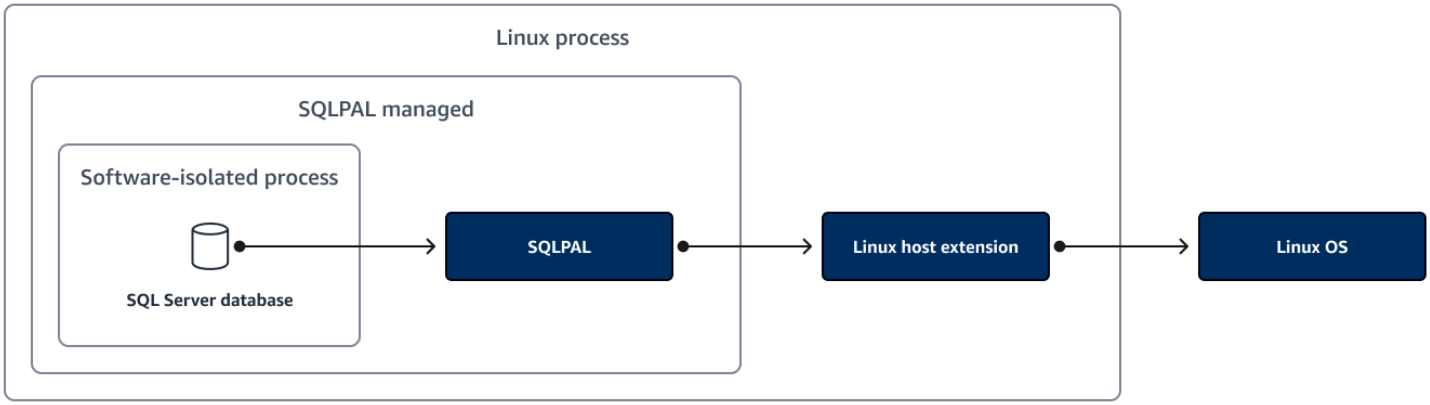
Linux 上の SQL Server を評価する

概要

SQL Server 2017 以降、SQL Server を Linux オペレーティングシステムにインストールできるようになりました。Linux 上の SQL Server はエンタープライズ対応であり、柔軟性、高パフォーマンス、セキュリティ機能、TCO の削減、HA/DR 機能、優れたユーザーエクスペリエンスを提供します。Windows Server の SQL Server から Linux の SQL Server に切り替えることで、Windows Server のライセンスコストを節約できます。

Linux の場合、SQL Server は、Red Hat Enterprise Linux (RHEL)、SUSE Linux Enterprise Server (SLES)、Ubuntu、Amazon Linux 2 にデプロイできます。SQL Server データベースエンジンは、Windows Server でも Linux でも同じ方法で実行されますが、Linux を使用する場合は、特定のタスクにいくつかの基本的な変更があります。SQL Server Always On アプリケーションを Linux と Windows で実行する場合の主な違いの 1 つは、フェイルオーバークラスタリングに関連しています。Windows Server ホストに Always On 可用性グループをデプロイする場合は、フェイルオーバークラスタリングをサポートする組み込み機能として Windows Server フェイルオーバークラスタリング (WSFC) と Active Directory を利用できます。しかし、WSFC も Active Directory も Linux でのフェイルオーバークラスタリングをサポートできません。Linux で SQL Server のフェイルオーバークラスタリングを起動する場合は、AWS Launch Wizard で、ClusterLabs Pacemaker を使用して、Linux インスタンスでのクラスターのセットアップと SQL のインストールを簡素化できます。

Windows 上および Linux 上の SQL Server は、共通のコードベースを共有します。つまり、SQL Server コアエンジンは、Linux 上で実行するための変更はまったく加えられていません。次の図に示すように、SQL Server はプラットフォーム抽象化レイヤー (SQLPAL) を導入しました。



SQLPAL は、SQL Server と基盤となるオペレーティングシステム間の呼び出しと通信の抽象化を担当します。ホスト拡張機能は、単なるネイティブ Linux アプリケーションです。低レベルのオペレーティングシステム関数は、I/O、メモリ、CPU 使用率を最適化するためのネイティブ呼び出しです。ホスト拡張機能が起動すると、SQLPAL をロードして初期化し、SQL Server を起動します。SQLPAL は、残りのコードに必要な変換を提供する分離したソフトウェアプロセスを起動します。この新しいレイヤーを SQL Server アーキテクチャに追加すると、Windows 上で SQL Server を強化しているのと同じエンタープライズレベルのコア機能と利点が、オペレーティングシステムに関係なく利用できるようになります。

コストへの影響

r5.2xlarge インスタンスの場合、Windows Server ライセンスのコスト削減は、各シナリオで約 268 USD です。この削減は、より安価な SQL Server エディションを使用する場合と比較して、サーバーの合計コストの割合が高くなります。次の表は、コスト削減を示しています。

インスタンス	Edition	Windows 上の SQL Server の月額コスト	Linux 上の SQL Server の月額コスト	削減量
r5.2xlarge	Web	735 USD	466 USD	37%
r5.2xlarge	標準	1,337 USD	1,068 USD	20%
r5.2xlarge	Enterprise	2,826 USD	2,558 USD	10%

> **Note**
>
> 上記の表の料金見積もりは、`us-east-1` リージョンのオンデマンド料金に基づいており、AWS 料金見積りツール で直接表示できます。

SMB セグメントの ISV 顧客が開発環境のコストを削減しようとしているシナリオの例を考えてみましょう。Windows サーバーのセットで SQL Server Developer Edition を既に使用しています。Windows の SQL Server Developer Edition から Linux の SQL Server Developer Edition に切り替えることで、ISV 顧客は開発ワークロードを 33% 削減できます。次の表は、このシナリオの推定コストを示しています。

Estimate	月額コスト
Windows + SQL Server	9,307.72 USD
Linux + SQL Server	6,218.36 USD
推定コスト削減額	3,089.36 USD (33%)

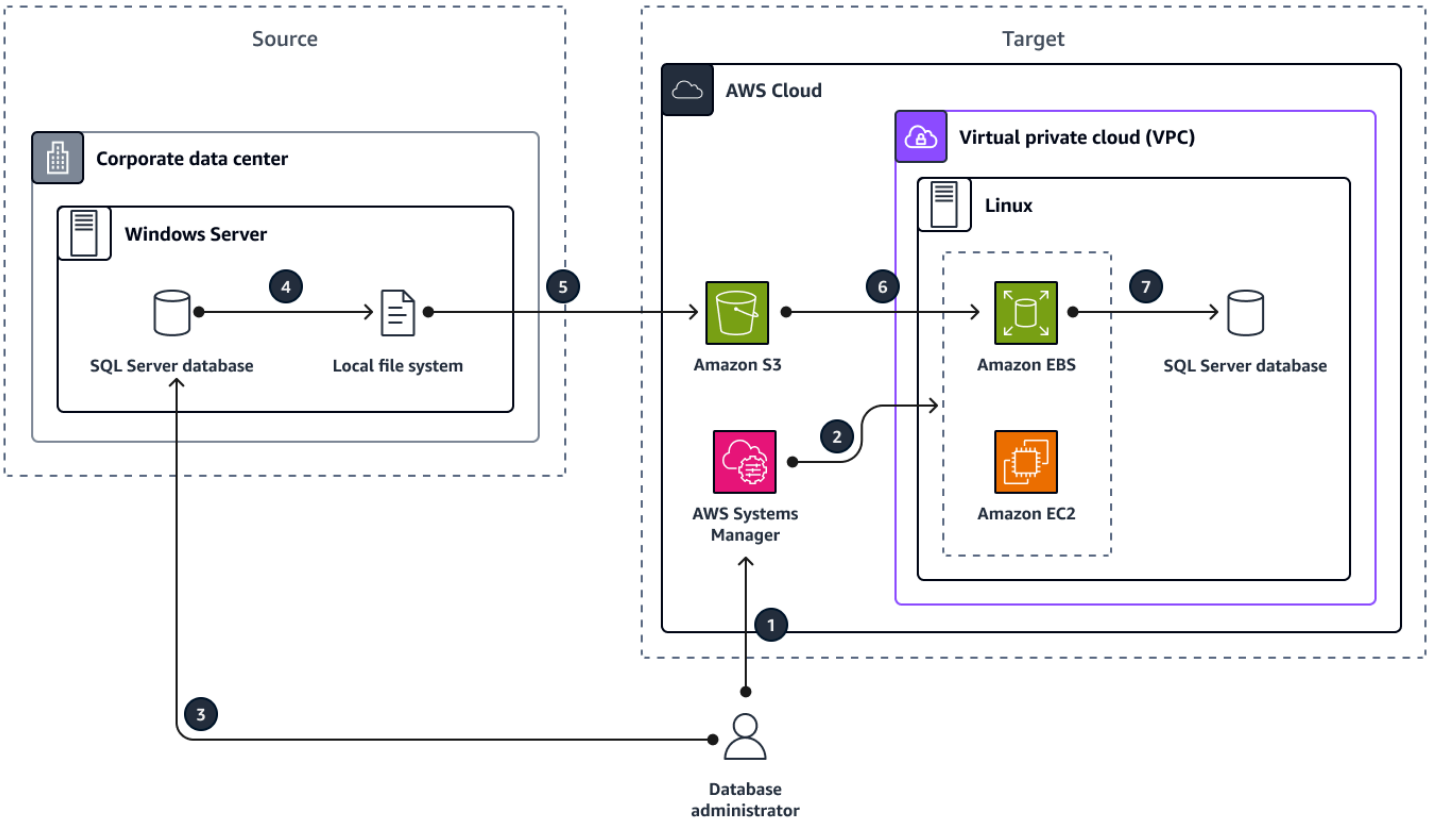
別のシナリオ例では、ある企業がライセンス込みの SQL Server EC2 インスタンスを Windows から Linux に移行します。同社は、Windows Server のライセンスコストを年間合計 300,000 USD 削減しています。これは合計 AWS 請求額の約 20% です。

コスト最適化の推奨事項

以下を検討することをお勧めします。

- Linux 上の SQL Server は、SQL Server 2017 以降でサポートされています。
- 切り替えを行うには、Microsoft SQL Server データベースの Windows から Linux へのリプラットフォームアシスタントを使用できます。リプラットフォームアシスタントは、一般的な非互換性をチェックし、Windows ホストからデータベースをエクスポートし、Ubuntu 16.04 で Microsoft SQL Server 2017 を実行している EC2 インスタンスにデータベースをインポートすることで、既存の SQL Server ワークロードを Windows オペレーティングシステムから Linux オペレーティングシステムに移行するのを支援するスクリプトツールです。
- SQL Server のバックアップおよび復元機能を使用して、Windows 上の SQL Server から Linux に切り替えることもできます。
- AWS Launch Wizard を使用すると、Linux または Ubuntu 上の SQL Server に簡単かつ迅速にデプロイできます。Launch Wizard では、アプリケーションのニーズに基づいて、スタンドアロンシナリオと高可用性シナリオの両方で SQL Server を Linux または Ubuntu にデプロイできます。詳細については、 AWS ブログの「Microsoft Workloads」の「Deploying to SQL Server Always on Linux with AWS Launch Wizard post」を参照してください。

次の図は、Microsoft SQL Server データベースの Windows から Linux へのリプラットフォームアシスタントを使用するソリューションのアーキテクチャを示しています。



その他のリソース

- 「Overview of SQL Server on Linux」(Microsoft Learn)
- 「Installation guide for SQL Server on Linux」(Microsoft Learn)
- を使用した SQL Server Always on Linux へのデプロイ AWS Launch Wizard (Microsoft Workloads on AWS Blog)
- Linux での高可用性 SQL Server (AWS オープンソースブログ)

SQL Server バックアップ戦略を最適化する

概要

ほとんどの組織は、SQL Server 上のデータを Amazon EC2 に保護し、前回のバックアップからの最大許容時間である目標復旧時点 (RPO) とサービス中断からサービス復旧までの最大許容遅延時間である目標復旧時間 (RTO) に関する現在の要件を満たす、適切なソリューションを求めています。EC2 インスタンスで SQL Server を実行している場合、データのバックアップを作成したり、

データを復元したりする複数のオプションがあります。SQL Server on Amazon EC2 でデータを保護するバックアップ戦略には、次のようなものがあります。

- Windows Volume Shadow Copy Service (VSS)-enabled Amazon Elastic Block Store (Amazon EBS) スナップショットまたは AWS Backup を使用するサーバーレベルのバックアップ
- SQL Server のネイティブバックアップと復元を使用するデータベースレベルのバックアップ

データベースレベルのネイティブバックアップを選択した場合、以下のストレージオプションがあります。

- Amazon EBS ボリュームを使用したローカルバックアップ
- Amazon FSx for Windows File Server または Amazon FSx for NetApp ONTAP を使用したネットワークファイルシステムのバックアップ
- AWS Storage Gateway を使用した Amazon Simple Storage Service (Amazon S3) へのネットワークバックアップ
- SQL Server 2022 の Amazon S3 への直接バックアップ

このセクションでは次のことを行います。

- ストレージスペースを節約するのに役立つ機能に焦点を当てる
- さまざまなバックエンドストレージオプション間のコストを比較する
- これらの推奨事項の実装に役立つ詳細なドキュメントへのリンクを提供する

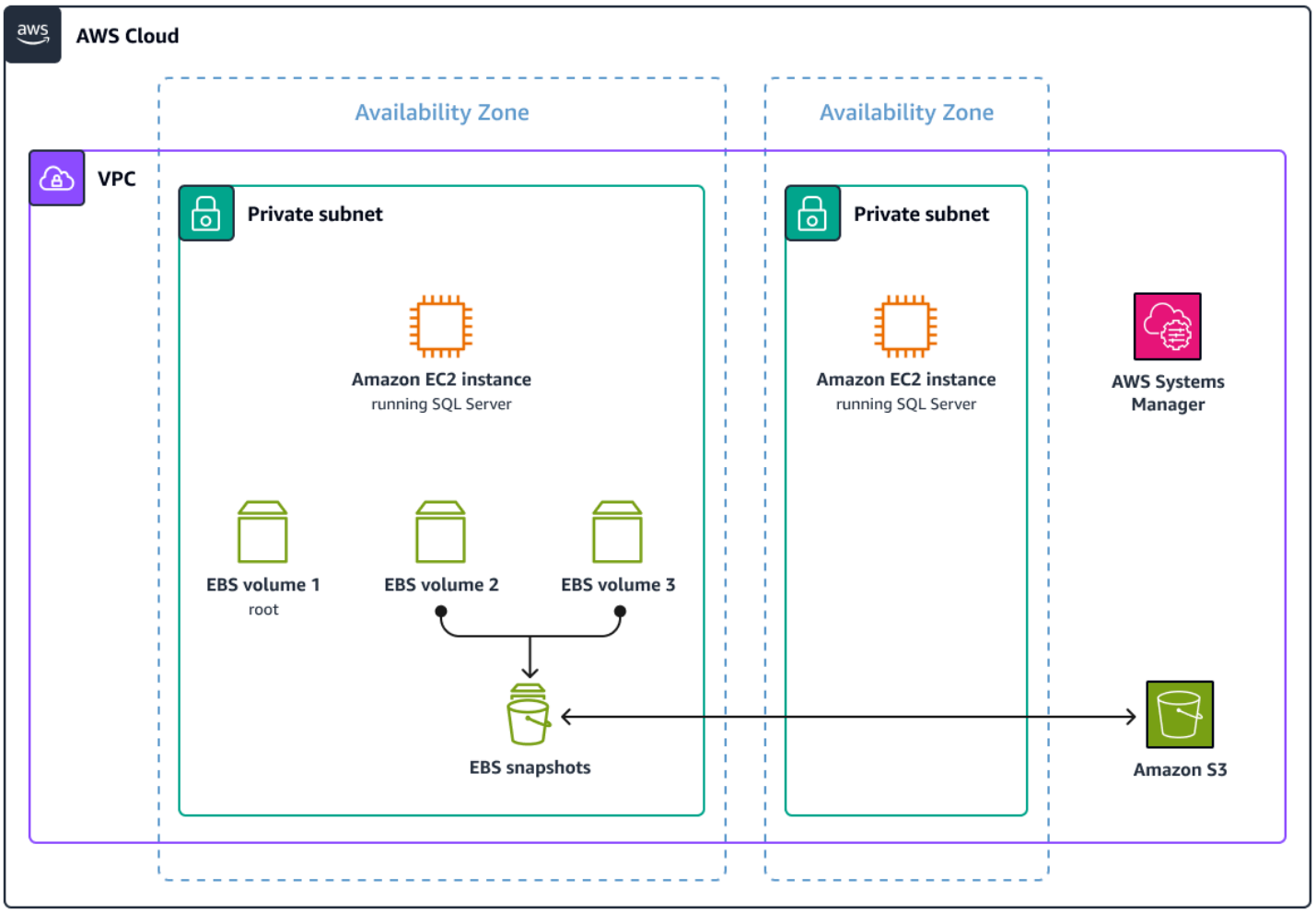
VSS 対応スナップショットを使用したサーバーレベルのバックアップ

VSS 対応スナップショットアーキテクチャでは、 AWS Systems Manager Run コマンドを使用して、SQL Server インスタンスに VSS エージェントをインストールします。この Run Command は、オペレーティングシステムとアプリケーションのバッファをディスクにフラッシュし、I/O オペレーションを一時停止し、EBS ボリュームのポイントインタイムスナップショットを取得して I/O を再開する、ワークフロー全体の呼び出しにも使用できます。

この Run Command は、ターゲットインスタンスにアタッチされたすべての EBS ボリュームの自動スナップショットを作成します。ユーザーデータベースファイルは、通常は他のボリュームに保存されるため、ルートボリュームを除外する選択もできます。複数の EBS ボリュームをストライピングして SQL Server ファイル用の 1 つのファイルシステムを作成する場合、Amazon EBS は、1 つの API コマンドで Crash-consistent なマルチボリュームのスナップショットもサポートします。アプリ

ケーション整合性のある VSS 対応 EBS スナップショットの詳細については、Amazon EC2 ドキュメントの「Create a VSS application-consistent snapshot」を参照してください。

次の図は、VSS 対応スナップショットを使用したサーバーレベルのバックアップのアーキテクチャを示しています。



VSS 対応スナップショットを使用する次の利点を考慮してください。

- DB インスタンスの初期のスナップショットには、フル DB インスタンスのデータが含まれています。同じ DB インスタンスの後続のスナップショットは増分です。つまり、直近のスナップショット以降に変更されたデータのみが保存されます。
- EBS スナップショットは、ポイントインタイムリカバリを提供します。
- スナップショットから、新しい SQL Server EC2 インスタンスに復元することができます。

- インスタンスが Amazon EBS を使用して暗号化されている場合、または、データベースが TDE を使用しているインスタンス内で暗号化されている場合、そのインスタンスまたはデータベースは、同じ暗号化を使用して自動的に復元されます。
- 使用している自動化されたクロスリージョンバックアップをコピーできます。
- EBS ボリュームをスナップショットから復元したとき、アプリケーションからすぐにアクセスすることができます。つまり、基盤となる 1 つ以上の EBS ボリュームをスナップショットから復元した後に、SQL Server をすぐにオンラインにすることができます。
- 復元したボリュームは、デフォルトでは、アプリケーションが初めてブロックを読み込む際に、Amazon S3 から基盤のブロックを取得します。これにより、EBS ボリュームをスナップショットから復元した後に、パフォーマンスの低下が生じる場合があります。ボリュームは、最終的には通常のパフォーマンスに追いつきます。ただし、このようなパフォーマンスの低下は、高速スナップショット復元 (FSR) のスナップショットを使うことで回避できます。
- EBS スナップショットのライフサイクル管理を使用できます。

VSS 対応スナップショットを使用する際には、次の制限事項を考慮してください。

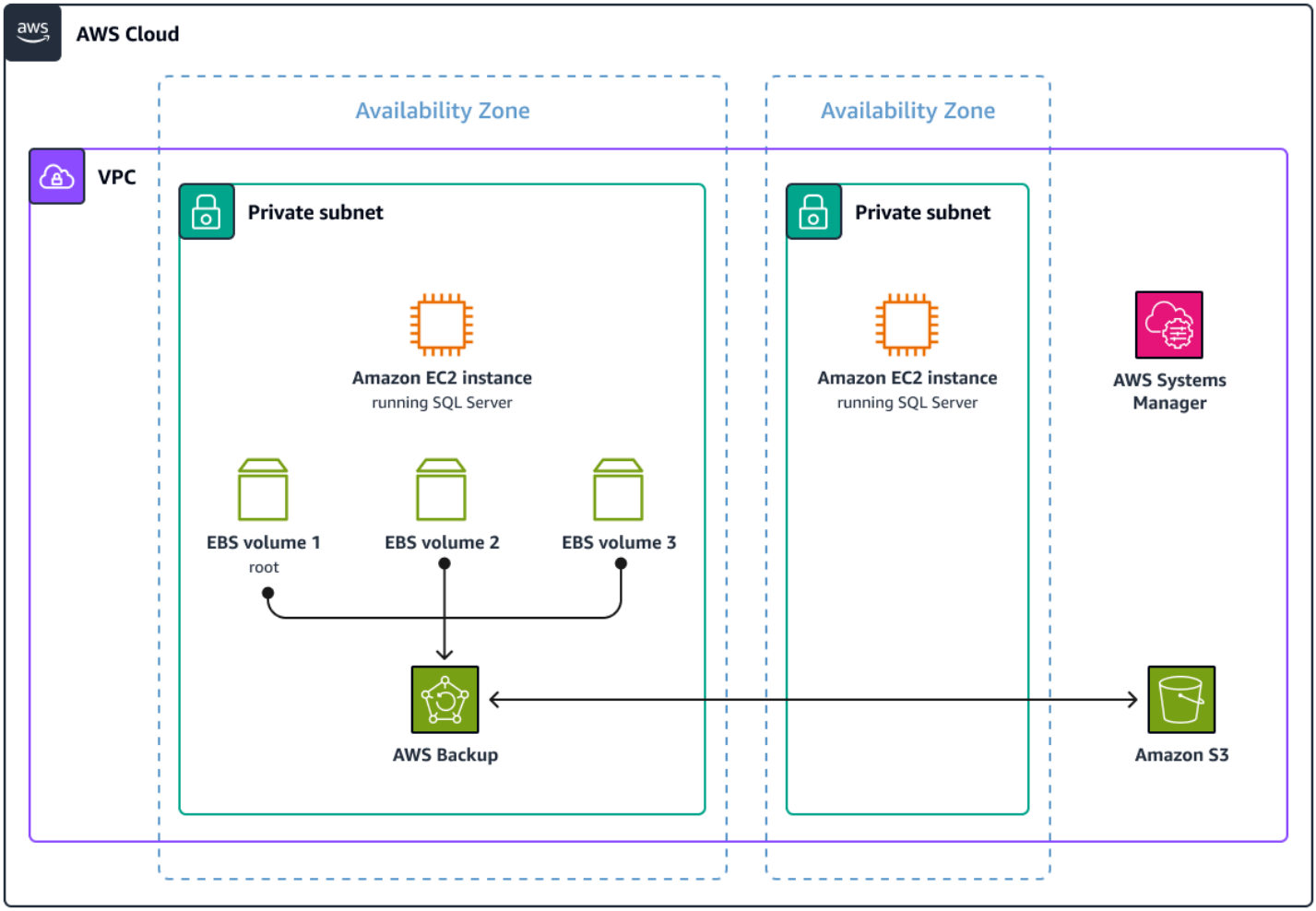
- SQL Server インスタンスの暗号化されたスナップショットでは、クロスリージョンのポイントインタイムリカバリを実行できません。
- 暗号化されていないインスタンスの、暗号化されたスナップショットを作成することはできません。
- スナップショットは EBS ボリュームレベルで作成されるため、個々のデータベースを復元することはできません。
- インスタンスをそれ自体に復元することはできません。
- DB インスタンスのスナップショットは、DB インスタンスと同じ AWS Key Management Service (AWS KMS) キーを使用して暗号化する必要があります。
- Storage I/O は、スナップショットのバックアッププロセス中に一瞬 (約 10 ミリ秒) 中断します。

を使用した SQL Server バックアップ AWS Backup

を使用してAWS Backupデータ保護を一元管理および自動化できます AWS のサービス。 AWS Backup は、大規模なデータ保護を簡素化する、費用対効果の高いフルマネージド型のポリシーベースのソリューションを提供します。 AWS Backup は、規制コンプライアンスの義務をサポートし、ビジネス継続性の目標を達成するのにも役立ちます。と を併用 AWS Backup すると AWS

Organizations、データ保護 (バックアップ) ポリシーを一元的にデプロイして、組織の AWS アカウント および リソース全体でバックアップアクティビティを設定、管理、管理できます。

以下の図は、 AWS Backupを使用した SQL Server on EC2 のバックアップと復元ソリューションのアーキテクチャを示したものです。



AWS Backupを使用して SQL Server をバックアップする次の利点を考慮してください。

- バックアップのスケジュール設定、保持の管理、ライフサイクル管理を自動化できます。
- バックアップ戦略を組織全体で一元化し、複数の アカウントと にまたがることができます AWS リージョン。
- AWS のサービス全体のバックアップアクティビティのモニタリングとアラートの発信を一元化できます。
- ディザスタリカバリ計画のリージョンを横断したバックアップを実装できます。
- このソリューションは、クロスアカウントバックアップをサポートしています。

- 二次的なバックアップ暗号化を使用した、安全なバックアップを実行できます。
- すべてのバックアップは、暗号化キーを使用した AWS KMS 暗号化をサポートしています。
- このソリューションは、TDE と連携しています。
- AWS Backup コンソールから復旧ポイントに復元することができます。
- SQL Server インスタンス全体をバックアップできます。これには、すべての SQL Server データベースが含まれます。

データベースレベルのバックアップ

以下のアプローチでは、Microsoft SQL Server ネイティブのバックアップ機能を使用します。SQL Server 上のインスタンスの、個々のデータベースのバックアップを作成し、個々のデータベースを復元することができます。

SQL Server ネイティブのバックアップと復元に使用されるオプションは、以下もサポートしています。

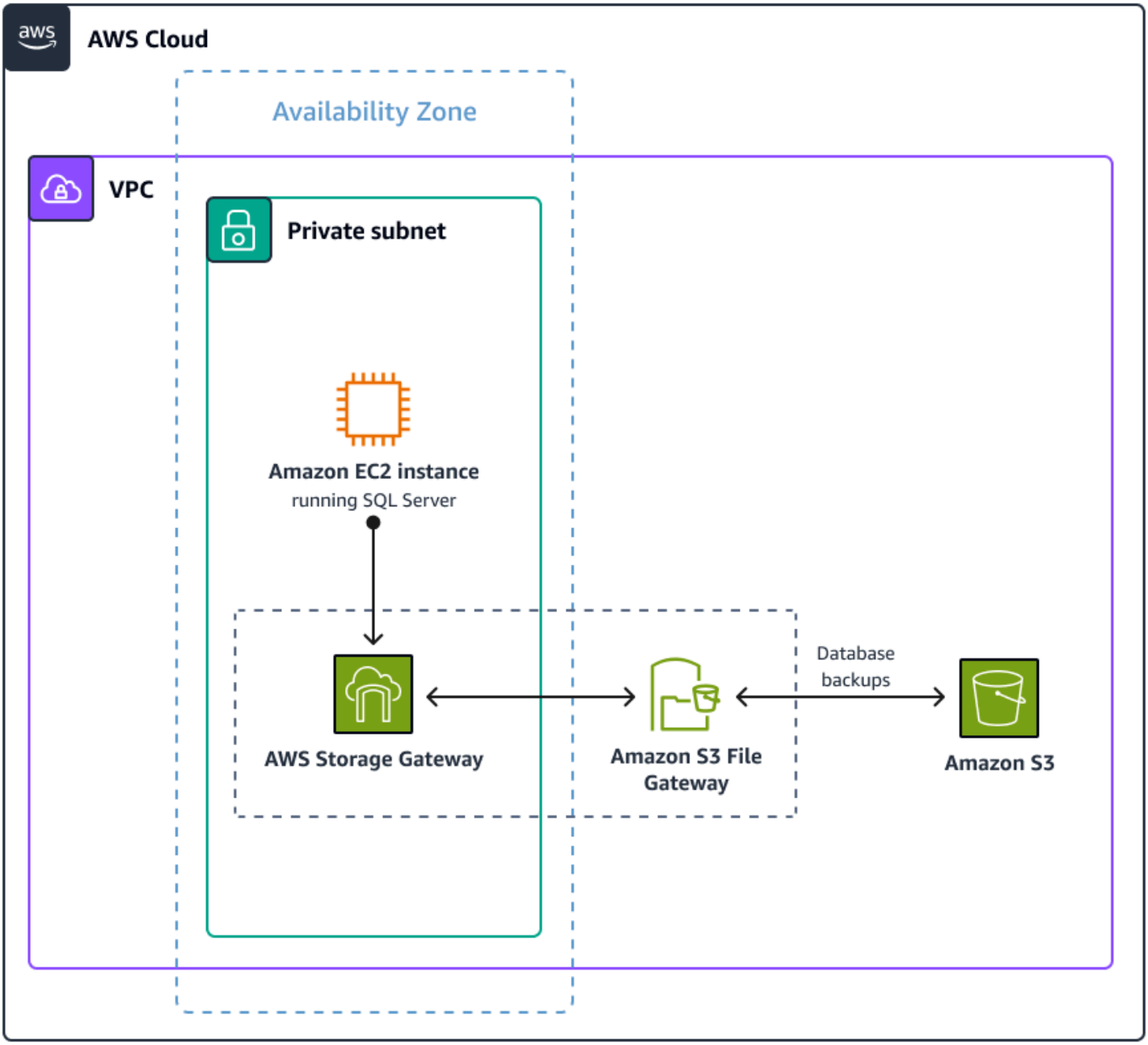
- 圧縮と複数ファイルのバックアップ
- フルバックアップ、差分バックアップ、T ログバックアップ
- TDE によって暗号化されたデータベース

SQL Server のネイティブバックアップと Amazon S3 への復元

SQL Server on Amazon EC2 は、SQL Server データベースのネイティブバックアップと復元をサポートしています。SQL Server データベースのバックアップを作成して、バックアップファイルを、既存のデータベースか、新しい SQL Server EC2 インスタンス、Amazon RDS for SQL Server、オンプレミスサーバーのいずれかに復元することができます。

Storage Gateway は、オンプレミスアプリケーションがクラウドストレージに実質的に無制限でアクセスできる、ハイブリッドクラウドのストレージサービスです。Storage Gateway を使用して Microsoft SQL Server データベースを Amazon S3 に直接バックアップすれば、オンプレミスのストレージフットプリントを減らすことができ、Amazon S3 を使用して耐久性と拡張性に優れた、費用対効果の高いストレージを実現することができます。

次の図は、Storage Gateway と Amazon S3 を使用したネイティブバックアップと復元のソリューションのアーキテクチャを示したものです。



Storage Gateway を使用した SQL Server ネイティブのバックアップを使用する次の利点を考慮してください。

- ストレージゲートウェイを、EC2 インスタンスのサーバーメッセージブロック (SMB) ファイル共有としてマッピングし、バックアップを Amazon S3 に送信できます。
- バックアップは S3 バケットに直接行われるか、Storage Gateway のファイルキャッシュを介して行われます。
- 複数ファイルのバックアップがサポートされています。

Storage Gateway を使用したネイティブバックアップの次の制限事項を考慮してください。

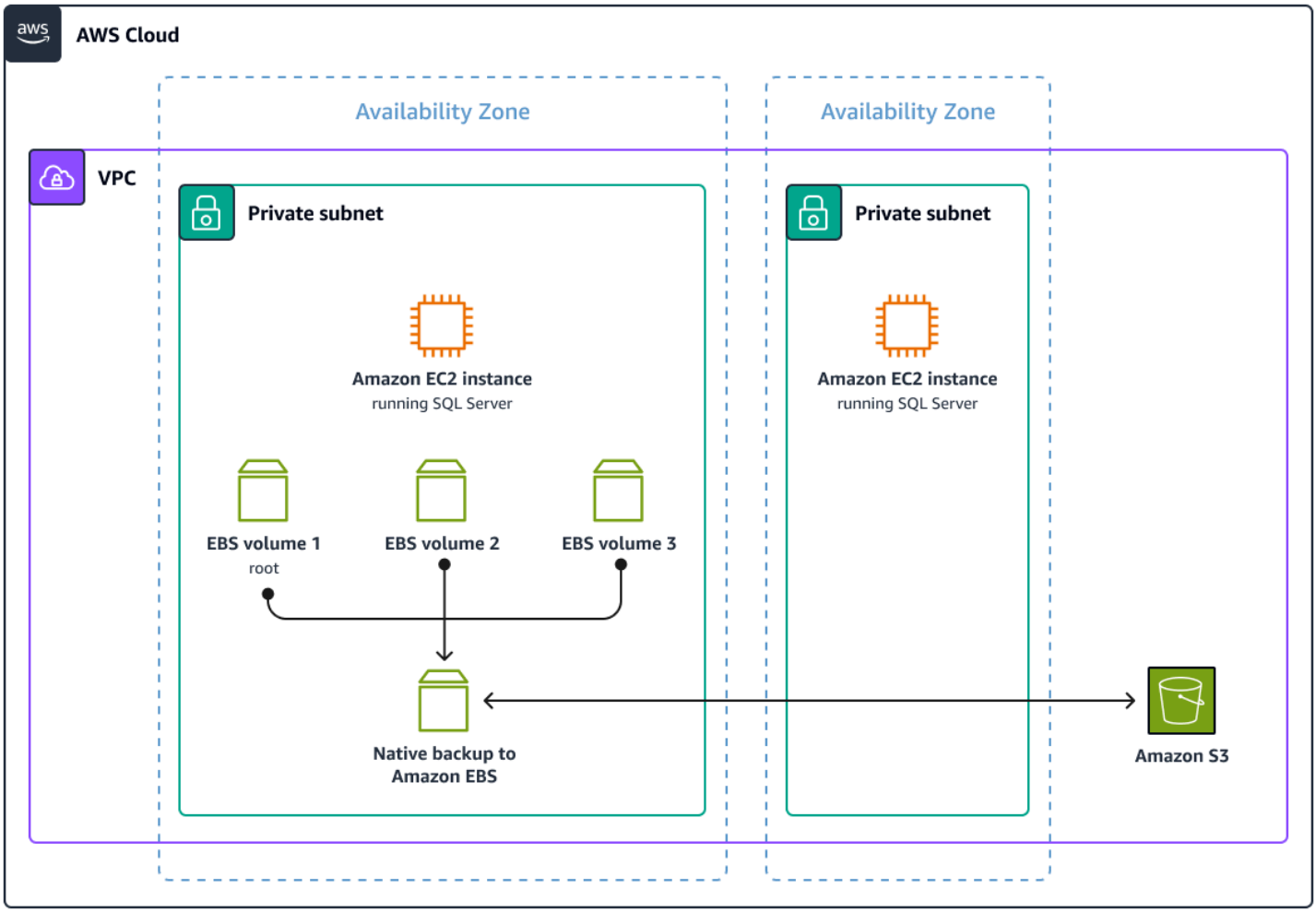
- バックアップと復元をデータベースごとに設定する必要があります。
- バックアップファイル用の Amazon S3 ライフサイクルポリシーの管理が必要になります。

Storage Gateway の設定方法の詳細については、 AWS ブログの記事「AWS Storage Gatewayを使用して SQL サーバーバックアップを Amazon S3 に保存する」を参照してください。

EBS ボリュームへの SQL Server のネイティブバックアップ

SQL Server データベースのネイティブバックアップを作成し、そのファイルを Amazon EBS ボリュームに保存することができます。Amazon EBS は、きわめて高性能なブロックストレージサービスです。EBS ボリュームは伸縮自在で、暗号化をサポートしています。デタッチして EC2 インスタンスにアタッチすることができます。EC2 インスタンス上の SQL Server は、同じ EBS ボリュームタイプで、または異なる EBS ボリュームタイプでバックアップすることができます。異なる EBS ボリュームにバックアップする利点の 1 つが、コストを削減できることです。

以下の図は、EBS ボリュームへのネイティブバックアップのアーキテクチャを示したものです。



EBS ボリュームへの SQL Server のネイティブバックアップを使用する次の利点を考慮してください。

- SQL Server の EC2 インスタンスにある個々のデータベースのバックアップを作成し、すべてのインスタンスを復元するのではなく個々のデータベースを復元することができます。
- 複数ファイルのバックアップがサポートされています。
- バックアップジョブを SQL Server エージェントと SQL Server ジョブエンジンを使用してスケジュールできます。
- 選択したハードウェアによって、パフォーマンス上の利点が得られます。例えば、st1 ストレージボリュームを使用するとスループットを高めることができます。

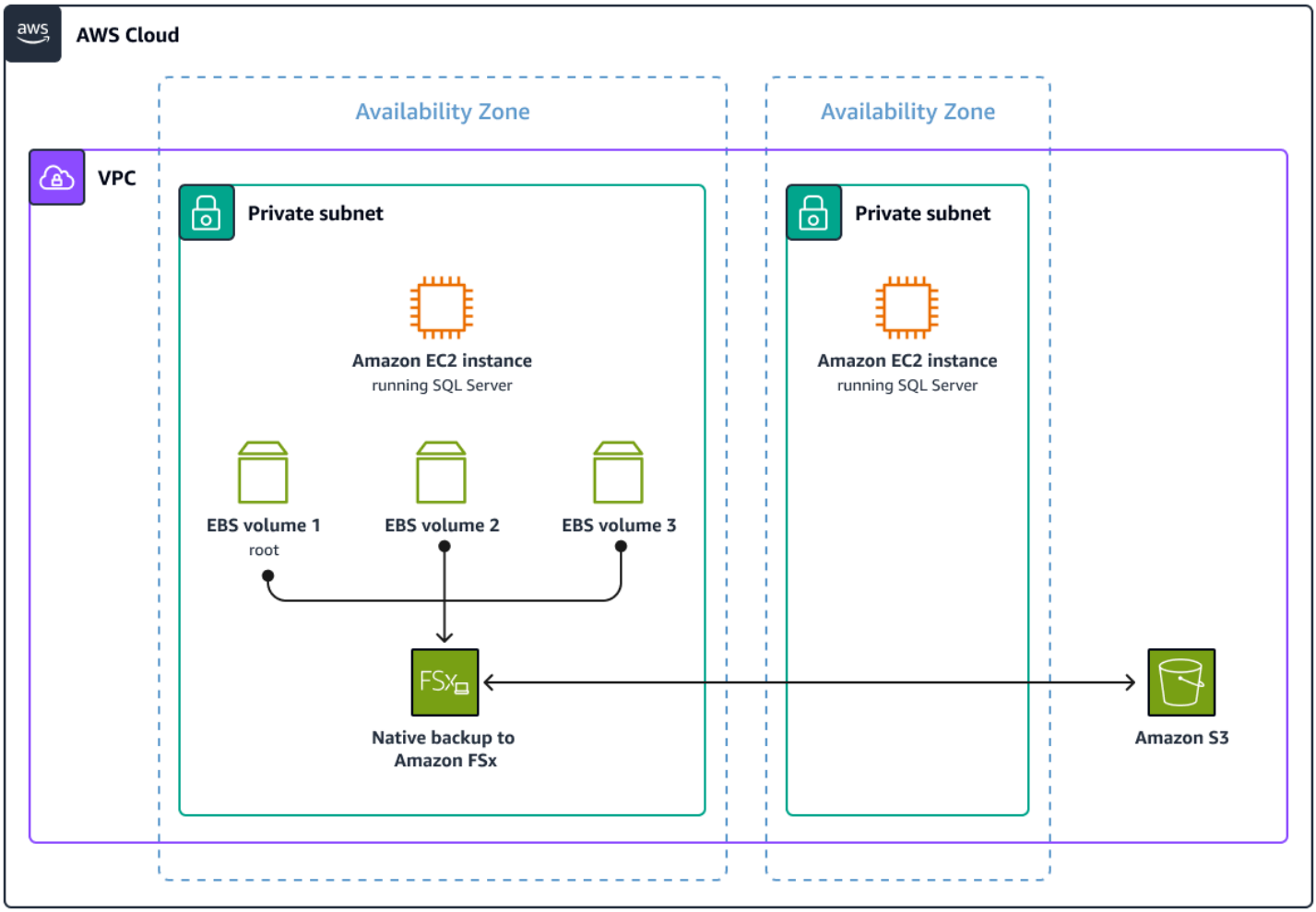
EBS ボリュームへのネイティブバックアップを使用する際には、次の制限事項を考慮してください。

- EBS ボリュームから Amazon S3 にバックアップを移動するときは、手動で行う必要があります。
- 大規模なバックアップの場合、Amazon EC2 のディスク容量を管理する必要があります。
- EC2 インスタンスでは、Amazon EBS のスループットがボトルネックになる可能性があります。
- Amazon EBS にバックアップを保存するには追加のストレージが必要になります。

Amazon FSx for Windows File Server への SQL Server ネイティブバックアップ

Amazon FSx for Windows File Server は、FSx for Windows File Server でのマルチ AZ ファイルシステムデプロイのネイティブサポート AWS を導入し、高速で予測可能で一貫したパフォーマンスを実現するように設計された、最大 64 TB のストレージを提供するフルマネージドのネイティブ Windows ファイルシステムです。ネイティブサポートにより、複数のアベイラビリティーゾーンを横断する高可用性と冗長性とを備えた AWS に、Windows ファイルストレージを簡単にデプロイできるようになりました。 AWS では、SMB Continuously Available (CA) ファイル共有のサポートも開始しました。FSx for Windows File Server を、SQL Server データベースのバックアップストレージとして使用できます。

以下の図は、FSx for Windows File Server への、SQL Server のネイティブバックアップのアーキテクチャを示したものです。



FSx for Windows File Server への SQL Server のネイティブバックアップを使用する次の利点を考慮してください。

- お使いの SQL Server データベースを Amazon FSx ファイル共有にバックアップすることができます。
- SQL Server インスタンスにある個々のデータベースのバックアップを作成し、すべてのインスタンスを復元するのではなく個々のデータベースを復元することができます。
- 複数の要素からなるバックアップがサポートされています。
- バックアップジョブを SQL Server エージェントとそのジョブエンジンを使用してスケジュールできます。
- インスタンスで Amazon EBS よりも高いネットワーク帯域幅を利用できます。

FSx for Windows File Server への SQL Server のネイティブバックアップを使用する際には、次の制限事項を考慮してください。

- AWS Backup または を使用して、Amazon FSx から Amazon S3 にバックアップを手動で移動する必要があります AWS DataSync。 FSx
- 大規模なバックアップでは、Amazon FSx のディスク容量の管理のため、追加のオーバーヘッドが必要になる場合があります。
- EC2 インスタンスのネットワークスループットがボトルネックになる可能性があります。
- FSx for Windows File Server にバックアップを保存するには追加のストレージが必要になります。

Amazon FSx for NetApp ONTAP への SQL Server のバックアップ

FSx for ONTAP のスナップショットは常に Crash-consistent ですが、アプリケーション整合性のあるスナップショットを作成するには、データベースを休止 (または I/O を一時停止) する必要があります。NetApp SnapCenter (SQL Server を含む特定のアプリケーションのプラグインを備えたオーケストレーションツール) と FSx for ONTAP を使用して、アプリケーション整合性のあるスナップショットを作成し、データベースを追加料金なしで保護、レプリケート、およびクローンすることができます。

NetApp SnapCenter

NetApp SnapCenter は、アプリケーション整合性のあるデータ保護のための統合プラットフォームです。SnapCenter では、スナップショットをバックアップと呼びます。本ガイドでは、この同じ命名規則を採用しています。SnapCenter には、アプリケーション整合性のあるバックアップ、復元、クローンを管理するための一元的なウィンドウが用意されています。特定のデータベースアプリケーションに SnapCenter プラグインを追加して、アプリケーション整合性のあるバックアップを作成します。SQL Server 用 SnapCenter プラグインには、データ保護ワークフローを簡素化する次の機能があります。

- 完全バックアップとログバックアップの粒度を備えたバックアップおよび復元オプション
- インプレース復元と別の場所への復元

SnapCenter の詳細については、 AWS ストレージブログの「Amazon FSx for NetApp ONTAP で NetApp SnapCenter を使用して SQL Server ワークロードを保護する」の投稿を参照してください。

バックアップのコスト最適化

次のオプションは、 AWSに SQL Server のバックアップを保存するコストを削減するのに役立ちます。

- バックアップファイルの作成中に SQL Server の圧縮を有効にし、可能な限り小さいファイルをストレージに送信します。例えば、3:1 の圧縮率は、ディスクスペースを約 66% 節約することを示しています。これらの列に対してクエリを実行するには、次の Transact-SQL ステートメントを使用できます: `SELECT backup_size/compressed_backup_size FROM msdb..backupset;`。
- S3 バケットにバックアップする場合は、Amazon S3 Intelligent-Tiering ストレージクラスを有効にして、ストレージコストを 30% 削減します。
- FSx for Windows File Server または FSx for ONTAP にバックアップする場合は、単一のアベイラビリティーゾーンを使用すると、複数のアベイラビリティーゾーンを使用する場合と比較して 50% のコスト削減になります。料金については、「Amazon FSx for Windows File Server の料金」と「Amazon FSx for NetApp ONTAP の料金」を参照してください。
- SQL Server 2022 の最も効率的なオプションは、Amazon S3 への直接バックアップです。Storage Gateway を回避することで、追加コストを節約できます。

バックアップのベンチマークテスト結果

このセクションでは、本ガイドで説明するバックアップソリューションのパフォーマンスベンチマークテストの結果に基づいて、サンプル 1 TB データベースのコストとパフォーマンスの観点から次のオプションを比較します。

- EC2 インスタンス仕様 – Windows Server 2019 および SQL Server 2019 Developer Edition を使用した r5d.8xlarge
- データベース仕様 – TDE が無効になっている 1 TB のサイズ

これらのテストは、r5d.8xlarge インスタンスと、1 TB の SQL Server データベースをソースとして使用して実行されました。ソースシステムはベストプラクティスに沿って構成され、ソースデータベースには、個別の gp3 ボリュームに分散された 4 つのデータファイル (各 250 GB) と 1 つのログファイル (50 GB) が含まれていました。SQL Server ネイティブの BACKUP コマンドには、圧縮を使ってバックアップパフォーマンスを最適化し、ネットワーク経由で送信されターゲットに書き込まれるデータの量を減らすため、10 のバックアップファイルへの書き込みが含まれています。すべてのテストケースで、ストレージのパフォーマンスがボトルネックでした。

この種のテストの考えられる構成は、無限に近いバリエーションがあります。このテストでは、パフォーマンス、コスト、スケーラビリティ、実際のユースケースの最適化に焦点を当てました。次の表は、バックアップターゲットオプションでキャプチャされたパフォーマンスメトリクスを示しています。

バックアップオプション	レベル	実行時間 (近似値)	バックアップレート	1 か月あたりのコスト (USD)*
ローカル EBS st1 HDD へのネイティブバックアップ、2 TB	データベース	00:30:46 分	554.7 Mbps	92.16 USD
ローカル EBS SSD gp3 へのネイティブバックアップ、2 TB	データベース	00:22:00 分	512 Mbps	193.84 USD
FSx for Windows File Server HDD へのネイティブバックアップ、2 TB @512 Mbps スループット	データベース	00:20:58 分	814.0 Mbps	1,146 USD
FSx for Windows File Server SSD へのネイティブバックアップ、2 TB @512 Mbps スループット	データベース	00:20:00 分	814.0 Mbps	1,326 USD
S3 ファイルゲートウェイ m6i.4xlarge (16 vCPU、64 GB) へのネイティブバックアップ、2 TB gp3	データベース	00:23:20 分	731.5 Mbps	470.42 USD
EBS VSS スナップショット	EBS ボリューム	00:00:02 秒 00:00:53 秒	スナップショットなし	51 USD

バックアップオプション	レベル	実行時間 (近似値)	バックアップレート	1 か月あたりのコスト (USD)*
AWS Backup (AMI バックアップ)	AMI	00:00:04 秒 00:08:00 分	スナップショットなし	75 USD
Amazon S3 への SQL Server の直接ネイティブバックアップ (SQL Server 2022)	データベース	00:12:00 分	731.5 Mbps	最初の 50 TB/月、1 GB あたり 0.023 USD、1 か月あたり 23.55 USD
FSx for ONTAP へのネイティブバックアップ (SnapCenter を使用)	データベース	–	–	440.20 USD

上記の表は、次のことを前提としています。

- データ転送と Amazon S3 のコストは含まれません。
- ストレージ料金はインスタンス料金に含まれます。
- コストは `us-east-1` リージョンに基づいています。
- 1 か月間の全体的な変化率が 10% である複数のバックアップにより、スループットと IOPS が 10% 増加します。

テスト結果は、最速のオプションが FSx for Windows File Server へのネイティブ SQL Server データベースバックアップであることを示しています。Storage Gateway とローカルにアタッチされた EBS ボリュームへのバックアップは、コスト効率の高いオプションですが、パフォーマンスが低下します。サーバーレベルのバックアップ (AMI) では、最適なパフォーマンス、コスト、管理性 AWS Backup を実現するために を使用することをお勧めします。

コスト最適化の推奨事項

Amazon EC2 における SQL Server のバックアップ用ソリューションについて理解しておくことは、データを保護し、バックアップのニーズを満たし、重大事象から回復するための計画を立てる上で鍵となります。このセクションで説明する、SQL Server インスタンスとデータベースをバックアップおよび復元するさまざまな方法は、データを保護し、組織の要件を満たすバックアップおよび復元戦略を考案するのに役立ちます。

このセクションでは、次のバックアップオプションについて説明します。

- Compression
- Amazon S3 の新しいストレージクラス、S3 Intelligent-Tiering を発表
- 単一のアベイラビリティーゾーン
- URL へのバックアップ

これらのオプションごとに提供されるガイダンスは高レベルです。組織でこれらの推奨事項のいずれかを実装する場合は、アカウントチームに連絡することをお勧めします。その後、チームは Microsoft スペシャリスト SA と連携して会話を主導することができます。optimize-microsoft@amazon.com に E メールを送信して連絡をとることもできます。

要約すると、次のことをお勧めします。

- SQL Server 2022 を使用している場合は、Amazon S3 へのバックアップが最もコスト効率の高いオプションです。
- SQL Server 2019 以前の SQL Server エディションを使用している場合は、最もコスト効率の高いオプションとして、Amazon S3 によって支援された Storage Gateway へのバックアップを検討してください。

Compression

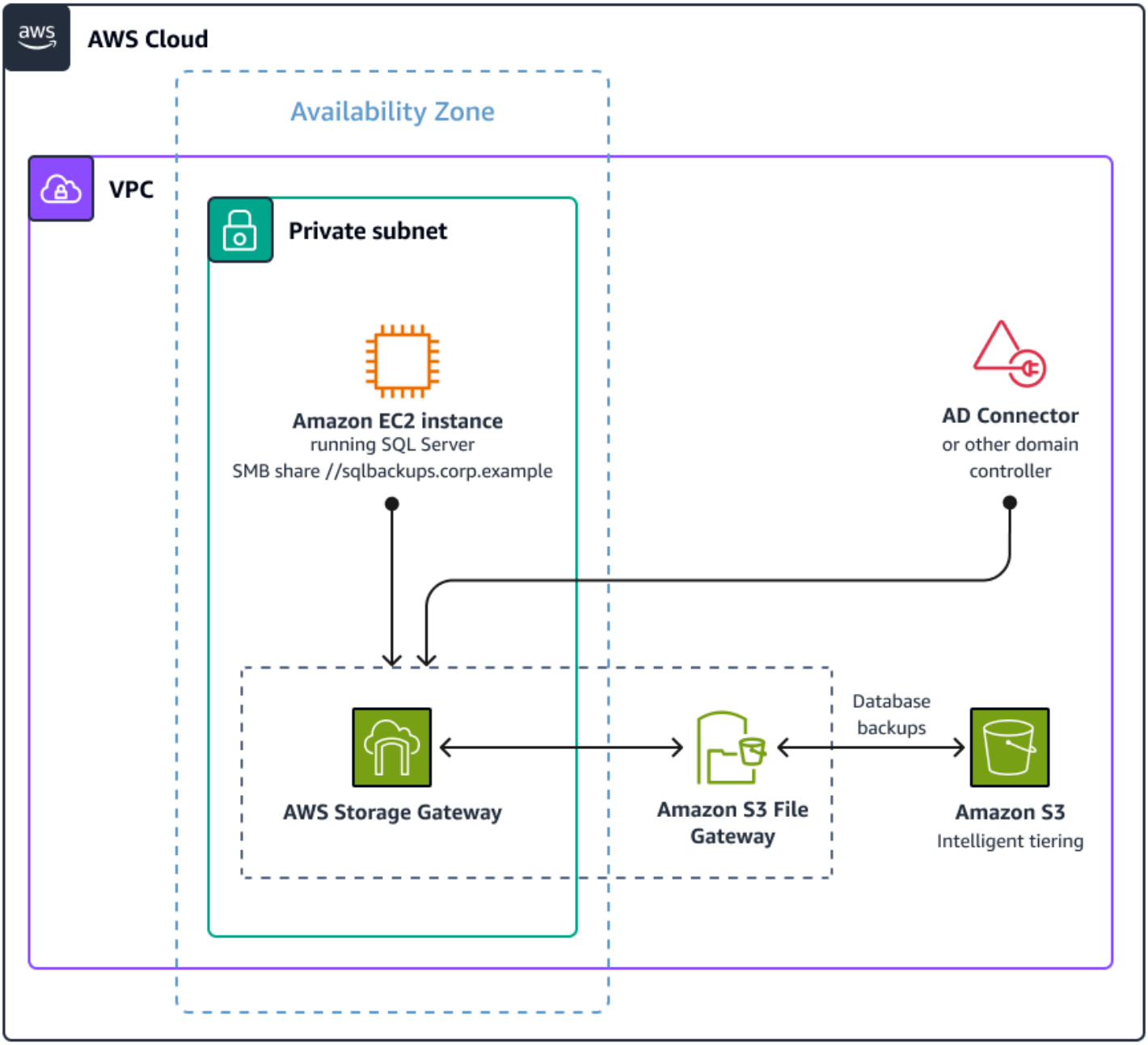
圧縮の目的は、バックアップごとに消費されるストレージを減らすことです。これは、さまざまなストレージオプションにとって有益です。SQL Server インスタンスのレベルで SQL Server バックアップの圧縮を有効にする必要があります。次の例は、バックアップデータベースで圧縮キーワードを追加する方法を示しています。

```
BACKUP DATABASE <database_name> TO DISK WITH COMPRESSION (ALGORITHM =
QAT_DEFLATE)
```

Amazon S3 の新しいストレージクラス、S3 Intelligent-Tiering を発表

Amazon S3 バケットへのバックアップの場合、Amazon S3 Intelligent-Tiering を Amazon S3 ファイルゲートウェイストレージクラスとして有効にできます。これにより、ストレージコストを最大 30% 削減できます。次に、Active Directory ドメインと統合できる SMB ファイル共有を使用して、S3 ファイルゲートウェイを SQL サーバーにマウントします。これにより、共有のアクセスコントロール、既存のサービスアカウントを活用する機能、一般的な Microsoft 向けファイルプロトコルを使用した Amazon S3 へのアクセスが可能になります。ドメインコントローラーに直接接続できない可能性があるアカウントでは、Active Directory Connector を使用して、オンプレミスまたはクラウドでの Active Directory との通信を容易にすることができます。ゲートウェイで Active Directory 設定を構成するには、ドメインコントローラーが Active Directory へのリクエストをプロキシするための Active Directory Connector IP を指定する必要があります。

次の図は、S3 Intelligent-Tiering に基づくソリューションのアーキテクチャを示しています。



デフォルトでは、S3 バケットに書き込まれたバックアップファイルは標準階層を使用します。バックアップファイルを標準階層から S3 Intelligent-Tiering に変換するには、ライフサイクルルールを作成する必要があります。AWS マネジメントコンソール を使用して S3 Intelligent-Tiering を有効にすることもできます。詳細については、 AWS ドキュメントの「Amazon S3 Intelligent-Tiering の使用を開始する」を参照してください。

単一のアベイラビリティーゾーン

単一のアベイラビリティーゾーンのファイルシステムを作成するには、FSx for Windows File Server ファイルシステムを作成するときにシングル AZ オプションを選択します。Amazon FSx は、Windows ボリュームシャドウコピーサービスを使用して、ファイルシステムの耐久性の高いバックアップ (Amazon S3 に保存) を毎日取得し、ユーザーはいつでも追加のバックアップを取ることができます。単一のアベイラビリティーゾーンの使用に関するいくつかの問題に注意してください。例えば、ファイルシステムがプロビジョニングされる、影響を受けるアベイラビリティーゾーンが一度に数時間ダウンすると、SMB ファイル共有にアクセスできなくなります。データへのアクセスが必要な場合は、ソースリージョン内の使用可能なアベイラビリティーゾーンのバックアップから復元する必要があります。詳細については、このガイドの「Use a single Availability Zone」セクションを参照してください。

URL へのバックアップ

SQL Server 2022 の場合、URL へのバックアップ機能を使用すると、Amazon S3 への直接バックアップが可能になります。これは、ストレージレイヤーで Amazon S3 の完全な機能セットを取得し、この機能を容易にするために以前のバージョンで必要なアプライアンスの AWS Storage Gateway コストを排除 AWS するため、 で実行されている SQL Server 2022 の理想的なバックアップアプローチです。この機能を実装する際に考慮すべき主なコストは、データ転送コストと、選択した S3 ストレージクラスの 2 つです。Amazon S3 のネイティブディザスタリカバリ機能が必要な場合は、クロスリージョンレプリケーションによってクロスリージョンのデータエグレスコストが発生することを考慮する必要があります。このオプションの設定方法の詳細については、Microsoft Workloads on AWS ブログの記事「Backup SQL Server databases to Amazon S3」を参照してください。

その他のリソース

- Amazon EC2 での SQL Server のバックアップと復元のオプション (AWS 規範ガイダンス)
- を使用した Amazon RDS Point-in-timeリカバリと継続的バックアップ AWS Backup (AWS ストレージブログ)
- Amazon FSx for NetApp ONTAP で NetApp SnapCenter を使用して SQL Server ワークロードを保護する (AWS ストレージブログ)
- Amazon S3 Intelligent-Tiering の使用開始 (AWS 入門リソースセンター)
- Amazon RDS for SQL Server のバックアップおよび復元戦略 (AWS データベースブログ)
- オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する (AWS 規範ガイダンス)

- Amazon EC2 に Microsoft SQL Server をデプロイするためのベストプラクティス (AWS ホワイトペーパー)

SQL Server データベースをモダナイズする

概要

スケーラビリティ、パフォーマンス、コスト最適化のためにレガシーデータベースをモダナイズするジャーニーを開始する場合、SQL Server などの商用データベースで課題に直面する可能性があります。商用データベースは高価で、顧客を閉じ込め、厳しいライセンス条項を課します。このセクションでは、SQL Server からオープンソースデータベースへの移行とモダナイズのオプションの概要を示し、ワークロードに最適なオプションの選択について説明します。

SQL Server データベースを Amazon Aurora PostgreSQL などのオープンソースデータベースにリファクタリングして、Windows および SQL Server のライセンスコストを節約できます。Aurora のようなクラウドネイティブの最新のデータベースは、オープンソースデータベースの柔軟性と低コストを、商用データベースの堅牢なエンタープライズグレードの機能と融合させます。可変ワークロードまたはマルチテナントワークロードがある場合は、Aurora Serverless V2 に移行することもできます。これにより、ワークロードの特性に応じてコストを最大 90% 削減できます。さらに、 AWS は Babelfish for Aurora PostgreSQL などの機能、 AWS Schema Conversion Tool (AWS SCT) などのツール、 AWS Database Migration Service (AWS DMS) などのサービスを提供し、SQL Server データベースの移行とモダナイゼーションを簡素化します AWS。

データベースサービス

Windows 上の SQL Server から、Amazon Aurora、Amazon RDS for MySQL、Amazon RDS for PostgreSQL などのオープンソースデータベースに移行すると、パフォーマンスや機能を損なうことなく大幅なコスト削減を実現できます。以下の点を考慮してください。

- Amazon EC2 上の SQL Server Enterprise Edition から Amazon RDS for PostgreSQL または Amazon RDS for MySQL に切り替えると、最大 80% のコスト削減につながります。
- Amazon EC2 上の SQL Server Enterprise Edition から Amazon Aurora PostgreSQL 互換エディションまたは Amazon Aurora MySQL 互換エディションに切り替えると、最大 70% のコスト削減につながります。

従来のデータベースワークロードの場合、Amazon RDS for PostgreSQL と Amazon RDS for MySQL は要件に対応し、リレーショナルデータベースにコスト効率の高いソリューションを提供しま

す。Aurora は、これまで高価な商用ベンダーに限定されていた多数の可用性およびパフォーマンス機能を追加します。Aurora の耐障害性機能は追加コストがかかります。ただし、他の商用ベンダーの同様の機能と比較すると、Aurora の耐障害性コストは、同じタイプの機能に対して商用ソフトウェアが請求するコストよりも依然として安価です。Aurora アーキテクチャは、標準の MySQL および PostgreSQL デプロイと比較してパフォーマンスが大幅に向上するように最適化されています。

Aurora はオープンソースの PostgreSQL および MySQL データベースと互換性があるため、移植性という追加の利点があります。最適なオプションが Amazon RDS for PostgreSQL、Amazon RDS for MySQL、または Aurora のいずれであっても、ビジネス要件を理解し、必要な機能を最適なオプションにマッピングする必要があります。

Amazon RDS と Aurora の比較

次の表は、Amazon RDS と Amazon Aurora の主な違いをまとめたものです。

Category	Amazon RDS for PostgreSQL または Amazon RDS for MySQL	Aurora PostgreSQL または Aurora MySQL
パフォーマンス	良いパフォーマンス	3 倍以上優れたパフォーマンス
フェイルオーバー	通常 60 ~ 120 秒*	通常 30 秒
スケーラビリティ	最大 5 個のリードレプリカ 秒単位の遅延	最大 15 個のリードレプリカ ミリ秒単位の遅延
Storage	最大 64 TB	最大 128 TB
ストレージ HA	1 つまたは 2 つのスタンバイを備えたマルチ AZ、それぞれデータベースコピーあり	デフォルトで 3 つのアベイラビリティーゾーンにまたがる 6 つのデータコピー
バックアップ	日次スナップショットとログのバックアップ	Amazon S3 への継続的な非同期バックアップ
Aurora でのイノベーション	NA	100 GB

Category	Amazon RDS for PostgreSQL または Amazon RDS for MySQL	Aurora PostgreSQL または Aurora MySQL
		データベースの高速クローン作成
	リードレプリカの自動スケーリング	
	クエリプラン管理	
	Aurora Serverless	
	グローバルデータベースによるクロスリージョンレプリカ	
	クラスターキャッシュ管理**	
	パラレルクエリ	
	データベースアクティビティストリーミング	

*大量のトランザクションがあると、フェイルオーバー時間が長くなる可能性があります。

**Aurora PostgreSQL で利用可能です。

次の表は、このセクションで説明するさまざまなデータベースサービスの推定月額コストを示しています。

データベースサービス	1 か月あたりのコスト (USD)*	AWS 料金見積りツール (必須 AWS アカウント)
Amazon RDS for SQL Server Enterprise Edition	3,750 USD	Estimate
Amazon RDS for SQL Server Standard Edition	2,318 USD	Estimate

データベースサービス	1 か月あたりのコスト (USD)*	AWS 料金見積りツール (必須 AWS アカウント)
Amazon EC2 の SQL Server Enterprise Edition	2,835 USD	Estimate
Amazon EC2 の SQL Server Standard Edition	1,345 USD	Estimate
Amazon RDS for PostgreSQL	742 USD	Estimate
Amazon RDS for MySQL	712 USD	Estimate
Aurora PostgreSQL	1,032 USD	Estimate
Aurora MySQL	1,031 USD	Estimate

* ストレージ料金はインスタンス料金に含まれます。コストは `us-east-1` リージョンに基づいています。スループットと IOPS は想定です。計算は r6i.2xlarge インスタンスと r6g.2xlarge インスタンスに対して行われます。

コスト最適化の推奨事項

通常、異種のデータベース間の移行では、データベーススキーマをソースからターゲットデータベースエンジンに変換し、ソースからターゲットデータベースにデータを移行する必要があります。移行の最初のステップは、SQL Server スキーマとコードオブジェクトを評価して、ターゲットデータベースエンジンに変換することです。

AWS Schema Conversion Tool (AWS SCT) を使用して、Amazon RDS for MySQL や Amazon RDS for PostgreSQL、Aurora MySQL、PostgreSQL などのさまざまなターゲットオープンソースデータベースオプションとの互換性についてデータベースを評価できます。Babelfish for Aurora PostgreSQL との互換性を評価するには、Babelfish Compass ツールを使用することもできます。これにより、 AWS SCT と Compass の強力なツールは、移行戦略を決定する前に、関連する先行作業を理解できます。続行する場合は、 AWS SCT がスキーマに必要な変更を自動化します。Babelfish Compass の中核となる哲学は、SQL データベースをまったく変更せずに、またはほとんど変更せずに Aurora に移行できるようにすることです。Compass は既存の SQL データベースを評価して、これを実現できるかどうかを判断します。これにより、SQL Server から Aurora へのデータの移行に労力を費やす前に、結果が明らかになります。

AWS SCT は、データベーススキーマとコードのターゲットデータベースエンジンへの変換と移行を自動化します。Babelfish for Aurora PostgreSQL を使用すると、スキーマの変更なしで、または最小限のスキーマ変更で、データベースとアプリケーションを SQL Server から Aurora PostgreSQL に移行できます。これにより、移行が加速されます。

スキーマの移行後、 AWS DMS を使用してデータを移行できます。 AWS DMS は、完全なデータロードを実行し、変更をレプリケートして、最小限のダウンタイムで移行を実行できます。

このセクションでは、次のツールについて詳しく説明します。

- AWS Schema Conversion Tool
- Babelfish for Aurora PostgreSQL
- Babelfish Compass
- AWS Database Migration Service

AWS Schema Conversion Tool

AWS SCT を使用して既存の SQL Server データベースを評価し、Amazon RDS または Aurora との互換性を評価できます。移行プロセスを簡素化するために、 AWS SCT を使用して、異種データベース移行でスキーマをあるデータベースエンジンから別のデータベースエンジンに変換することもできます。 AWS SCT を使用してアプリケーションを評価し、C#、C++、Java、およびその他の言語で記述されたアプリケーションの埋め込みアプリケーションコードを変換できます。詳細については、 AWS SCT ドキュメントの「AWS SCTを使用したアプリケーション SQL の変換」を参照してください。

AWS SCT は、多くのデータベースソースをサポートする無料の AWS ツールです。を使用するには AWS SCT、ソースデータベースをポイントし、評価を実行します。次に、AWS SCT がスキーマを評価し、評価レポートを生成します。評価レポートには、エグゼクティブサマリー、複雑さと移行作業、適したターゲットデータベースエンジン、変換に関する推奨事項が含まれます。ダウンロードするには AWS SCT、 ドキュメントの「インストール、検証、更新 AWS SCT」を参照してください。AWS SCT

次の表は、データベースを異なるターゲットプラットフォームに変更する際の複雑さを示すために AWS SCT によって生成されるエグゼクティブサマリーの例を示しています。

ターゲットプラッ	自動または最小限の変更	複雑なアクション

トフォーム

	ストレージオブジェクト	コードオブジェクト	変換アクション	ストレージオブジェクト		コードオブジェクト	
Amazon RDS for MySQL	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56
Amazon Aurora MySQL 互換エディション	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56
Amazon RDS for PostgreSQL	60 (98%)	12 (52%)	54	1 (2%)	1	11 (48%)	26
Amazon Aurora PostgreSQL 互換エディション	60 (98%)	12 (52%)	54	1 (2%)	1	11 (48%)	26
Amazon RDS for MariaDB	60 (98%)	7 (30%)	42	1 (2%)	1	16 (70%)	58
Amazon Redshift	61 (100%)	9 (39%)	124	0 (0%)	0	14 (61%)	25
AWS Glue	0 (0%)	17 (100%)	0	0 (0%)	0	0 (0%)	0

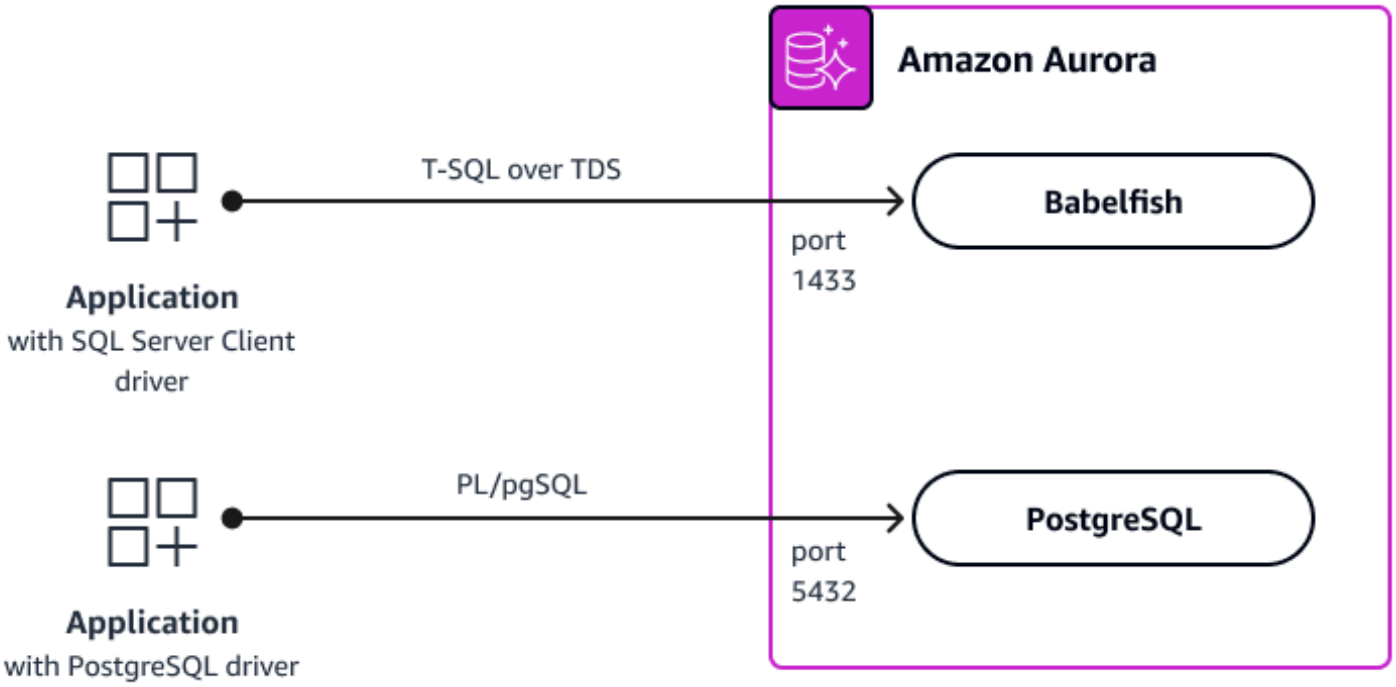
Babelfish	59 (97%)	10 (45%)	20	2 (3%)	2	12 (55%)	30

AWS SCT レポートには、自動的に変換できないスキーマ要素の詳細も表示されます。AWS 移行プレイブックを参照して、 AWS SCT 変換ギャップを埋め、ターゲットスキーマを最適化できます。異種移行を支援するデータベース移行プレイブックは多数あります。

Babelfish for Aurora PostgreSQL

Babelfish for Aurora PostgreSQL は、SQL Server クライアントからデータベース接続を受け入れる機能を使用して Aurora PostgreSQL を拡張します。Babelfish により、少ないコード変更で、またデータベースドライバを変更せずに、元々 SQL Server 用に構築されているアプリケーションを直接 Aurora PostgreSQL で利用できるようになります。Babelfish は Aurora PostgreSQL をバイリンガルに変換して、Aurora PostgreSQL が T-SQL 言語と PL/pgSQL 言語の両方を操作できるようにします。Babelfish は、SQL Server から Aurora PostgreSQL への移行作業を最小限に抑えます。これにより、移行が加速され、リスクが最小限に抑えられ、移行コストが大幅に削減されます。移行後も T-SQL を引き続き使用できますが、開発に PostgreSQL ネイティブツールを使用するオプションもあります。

次の図は、T-SQL を使用するアプリケーションが SQL Server のデフォルトポート 1433 に接続し、Babelfish トランスレーターを使用して Aurora PostgreSQL データベースと通信する方法を示しています。一方、PL/pgSQL を使用するアプリケーションは、Aurora PostgreSQL のデフォルトポート 5432 を使用して Aurora PostgreSQL データベースに直接および同時に接続できます。



Babelfish は、特定の SQL Server T-SQL 機能をサポートしていません。このため、Amazon は、SQL ステートメントを 1 行ずつ分析し、Babelfish でサポートされていないものがあるかどうかを判断するための評価ツールを提供しています。

Babelfish の評価には 2 つのオプションがあります。 は、SQL Server データベースと Babelfish の互換性を評価 AWS SCT できます。もう 1 つのオプションは Babelfish Compass ツールです。Compass ツールは Babelfish for Aurora PostgreSQL の新しいリリースに合わせて更新されるため、これが推奨されるソリューションです。

Babelfish Compass

Babelfish Compass は、Babelfish for Aurora PostgreSQL の最新リリースに対応した無料のダウンロード可能なツールです。対照的に、 AWS SCT はしばらくすると新しい Babelfish バージョンをサポートします。Babelfish Compass は、SQL Server データベーススキーマに対して実行されます。SQL Server Management Studio (SSMS) などのツールを使用して、ソース SQL Server データベーススキーマを抽出することもできます。その後、Babelfish Compass を使用してスキーマを実行できます。これにより、SQL Server スキーマと Babelfish の互換性、および移行前に変更が必要かどうかを詳述するレポートが生成されます。Babelfish Compass ツールは、これらの変更の多くを自動化し、最終的に移行を加速させることもできます。

評価と変更が完了したら、SSMS や sqlcmd などの SQL Server ネイティブツールを使用して、スキーマを Aurora PostgreSQL に移行できます。手順については、 AWS データベースブログの記事「Migrate from SQL Server to Amazon Aurora using Babelfish」を参照してください。

AWS Database Migration Service

スキーマの移行後、 AWS Database Migration Service (AWS DMS) を使用して最小限のダウンタイム AWS でデータを に移行できます。 は完全なデータロードを行う AWS DMS だけでなく、ソースシステムの稼働中にソースから宛先への変更をレプリケートします。ソースデータベースとターゲットデータベースの両方が同期された後、アプリケーションが移行を完了するターゲットデータベースを指す場合にカットオーバーアクティビティを実行できます。 AWS DMS は現在、Aurora PostgreSQL ターゲットに対して Babelfish でフルデータロードのみを実行し、変更をレプリケートしません。詳細については、 AWS DMS ドキュメントの「 のターゲットとして Babelfish を使用する AWS Database Migration Service」を参照してください。

AWS DMS は、同種 (同じデータベースエンジン間) 移行と異種 (異なるデータベースエンジン間) 移行の両方を実行できます。 は、多くのソースデータベースエンジンと宛先データベースエンジン AWS DMS をサポートしています。詳細については、 AWS データベースブログの投稿を使用して SQL Server データベースを Amazon RDS for SQL Server に移行する AWS DMSを参照してください。

その他のリソース

- さよなら Microsoft SQL Server、Hello Babelfish (AWS ニュースブログ)
- CLI を使用したデータベーススキーマとアプリケーション SQL の AWS Schema Conversion Tool 変換 (AWS データベースブログ)
- ベストプラクティスと フィールドから学んだ教訓を使用して SQL Server を Amazon Aurora PostgreSQL に移行する (AWS データベースブログ)
- Microsoft SQL Server から Amazon RDS for PostgreSQL および Amazon Aurora PostgreSQL への移行後のデータベースオブジェクトの検証 (AWS データベースブログ)

SQL Server のストレージを最適化する

概要

このセクションでは、EC2 ワークロード上の SQL Server 用 Amazon Elastic Block Store (Amazon EBS) SSD ストレージのコスト最適化に焦点を当てます。

SQL Server ワークロードをデプロイして実行するためのさまざまなストレージオプションがあります AWS。適切なストレージの選択は、目的、アーキテクチャ、耐久性、パフォーマンス、容量、コストに基づいて行う必要があります。SQL Server ワークロードを実行する AWS お客様は、通常、Amazon EBS、NVMe、Amazon FSx、Amazon Simple Storage Service (Amazon S3) ストレージの組み合わせを使用します。

Amazon EBS は、EC2 コンピューティングインスタンスに接続されたネットワークアタッチドストレージであり、一般的なオペレーティングシステム、アプリケーション、データベース、バックアップファイルの保存と処理に使用されます。Amazon EBS ソリッドステートドライブ (SSD) ストレージには、汎用 SSD (gp2 および gp3) とプロビジョンド IOPS SSD (io1、io2、io2BX) が含まれています。以下の点を考慮してください。

- r5d などの一部の EC2 インスタンスには、ホストインスタンスに物理的にアタッチされたローカル NVMe SSD があります。これらのボリュームは、一般的に SQL Server tempdb またはバッファプール拡張機能に使用されるブロックレベルストレージを提供します。
- Amazon FSx for Windows File Server はフルマネージド型のファイルストレージサービスであり、Amazon FSx for NetApp ONTAP は、NetApp の一般的な ONTAP ファイルシステム上に構築されたフルマネージド型の共有ストレージです。Amazon FSx は、高可用性の SQL Server フェイルオーバークラスターインスタンス (FCI) 設定で SQL Server ワークロードを実行するためによく使用されます。このソリューションは SQL Server のデータおよびログファイルをホストするため、EC2 インスタンスの EBS パフォーマンス要件が軽減されます。
- Amazon S3 は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。SQL Server のネイティブバックアップファイル、AMI、EBS スナップショット、アプリケーションログなどを Amazon S3 に保存できます。

Amazon EBS の SSD ストレージタイプ、パフォーマンス、コスト

Amazon EBS の SSD ストレージコストは、一般的に耐久性とパフォーマンスの向上に伴って増加します。現在、ストレージには 5 つのボリュームタイプがあり、それぞれに独自のパフォーマンスメトリクスがあります。SSD ベースのボリュームのユースケースと特性の概要については、Amazon EBS ドキュメントの「Solid state drive (SSD) volumes」セクションの表を参照してください。

Amazon CloudWatch を使用して、SSD のパフォーマンスのモニタリング、トレンドデータのキャプチャ、特定のしきい値に達したときのアラームの設定を行うことができます。 AWSで SQL Server ワークロードを実行している場合は、詳細なモニタリングを有効にして CloudWatch カスタムメトリクスをデプロイし、ディスクレイテンシー、IOPS、スループット、ディスクキューの長さ、使用済

みキャパシティと空きキャパシティなどの詳細なボリュームパフォーマンスメトリクスをキャプチャすることを検討してください。これらの CloudWatch パフォーマンスメトリクスを使用して、プロビジョニング不足およびプロビジョニング過剰のストレージを特定し、履歴データポイントを提供してストレージ要件を正確に定義できます。

Amazon EBS の SSD ストレージコストも、割り当てられたキャパシティによって異なります。次の表は、さまざまなボリュームタイプの比較を示しています。すべてのボリュームタイプに、1 TB のキャパシティと同様のパフォーマンス設定があります。

ボリュームタイプ	最大 IOPS (16 KiB I/O)	最大スループット (128 KiB I/O)	1TB あたりの料金	コスト削減率
gp2	3,000	250	102.40 USD	
gp3	3,000	250	86.92 USD	15%
io1	16,000	500	1,168 USD	
io2	16,000	500	1,168 USD	
gp3	16,000	500	146.92 USD	87%
io2bx	16,000	4,000	1,168 USD	
gp3	16,000	1,000	181.92 USD	84%

> **Note**
>
> 上記の表のパフォーマンスとコストのメトリクスは、 AWS 料金見積りツールからの見積もりに基づいたボリュームあたりのメトリクスです。の見積りにアクセスするには、 AWS アカウント が必要です AWS 料金見積りツール。

Amazon EBS SSD gp3 ボリュームは、低コストで優れたパフォーマンスを提供します。16,000 未満の IOPS、500 MiBps 未満のスループットを必要とするワークロードで、io1 または io2 ボリュームよりも gp3 ボリュームを選択した場合、最大 87% 節約できます。

io2 Block Express (io2BX) ボリュームは、通常の io2 ボリュームよりも優れたパフォーマンスを提供します。16,000 IOPS では、io1 または io2 ボリュームは 500 MiBps のスループットしか対応できま

せんが、io2BX ボリュームには最大 4,000 MiBps のスループットを設定できます。io1 および io2 ボリュームと比較すると、io2BX ボリュームは、まったく同じ料金で 16,000～64,000 IOPS で 4 倍を超えるスループットを提供します。通常の io2 ボリュームは、io2BX 対応の EC2 インスタンスにアタッチすることで、io2BX ボリュームに変換できます。io2BX 対応 EC2 インスタンスのリストについては、Amazon EBS ドキュメントの「Provisioned IOPS SSD ボリューム」を参照してください。新しいストレージをデプロイする前に、AWS 料金見積りツール を使用して毎月のコストを見積もり、耐久性、パフォーマンス、キャパシティのトレードオフに基づいてコストへの影響を理解することができます。

Amazon EBS の一般的な SSD コスト最適化

保存する内容を評価し、適切なストレージタイプとクラスを使用していることを確認することをお勧めします。例えば、Amazon S3 は、SQL Server バックアップに最適な優れたプライスポイント、組み込みライフサイクルポリシー、レプリケーションオプションを提供します。SQL Server 2022 は Amazon S3 に直接バックアップできますが、以前のバージョンの SQL Server はネイティブローカルバックアップに依存しています。古いバージョンの SQL Server を実行している場合は、Amazon EBS HDD ボリュームにバックアップしてから、そのバックアップを Amazon S3 にコピーすることを検討してください。このソリューションでは、バックアップに gp3 ボリュームを使用する場合と対照的に、53% 節約できます。

次の表は、Amazon EBS gp3、Amazon EBS HDD st1、および Amazon S3 上の 1 TB のストレージの料金差を示しています。

ストレージタイプ	Capacity	月額料金
EBS gp3 500 MiBps	1 TB	96.92 USD
EBS st1 バースト 500 MiBps		46.08 USD
S3 Standard		23.55 USD
S3 Standard (低頻度アクセス)		12.80 USD
S3 Glacier Deep Archive		1.03 USD

> **Note**
>
> 上記の表のコストのメトリクスは、 AWS 料金見積りツールの見積もりに基づいています。 の見積りにアクセスするには、 AWS アカウント が必要です AWS 料金見積りツール。

以下を検討することをお勧めします。

- 詳細なモニタリングを有効にし、CloudWatch カスタムメトリクスをデプロイして、ストレージのパフォーマンス要件を正確にキャプチャします。
- Amazon EBS ストレージを gp2 から gp3 にアップグレードして、コストを削減し、柔軟性を高め、パフォーマンスを向上させます。
- Amazon EBS ストレージを io1 から io2 にアップグレードして、耐久性とパフォーマンスの柔軟性を高めます。
- 耐久性とパフォーマンスを向上させるために、可能であれば io1 または io2 の代わりに io2BX を使用します。
- キャパシティ要件と高パフォーマンスボリュームのコストを削減するために、ストレージを選択するときは種々の組み合わせを検討してください。例えば、ルートボリューム (オペレーティングシステム)、SQL Server のインストール、システムデータベース (tempdb を除く)、およびパフォーマンスの低いユーザーデータベースには低コストの gp3 ボリュームを使用できます。これにより、io2 ボリュームのキャパシティとコストを削減できます。io2 ボリュームは、高パフォーマンスユーザーデータベース専用にすることができます。
- SQL Server データベースを でホストする場合は AWS、データベースごとに複数の SQL Server データファイルを使用することをお勧めします。これにより、読み取り/書き込みワークロードを複数のボリュームに分散できるため、ボリュームあたりのパフォーマンスとキャパシティの要件が軽減され、結果としてコストを削減できます。
- 本番ワークロードが io1 や io2/io2BX などの高パフォーマンスストレージを必要とする場合でも、コストを削減するために非本番ワークロードには gp3 ボリュームの使用を検討してください。
- ストレージ使用率を経時的に追跡および傾向分析して、使用量の急増や予期しないコストを容易に特定します。
- 実際の使用率に基づいて EBS ボリュームをスケールアップまたはスケールダウンするための推奨事項を取得するには AWS Compute Optimizer を使用します。
- の伸縮性を使用して AWS 、Amazon EBS の SSD ボリュームのパフォーマンスと容量のニーズを調整します。オンプレミス環境とは異なり、将来のワークロード用にストレージのパフォーマンスとキャパシティを過剰にプロビジョニングする必要はありません。データベースをオンラインに保

ちながら、既存の SQL Server ワークロードを に移行 AWS し、必要に応じてパフォーマンスや容量を調整できます。

その他のリソース

- 「Amazon EBS volume types」(Amazon EBS ドキュメント)
- 「Amazon Elastic Block Store (Amazon EBS)」(Amazon EBS ドキュメント)
- 「Provisioned IOPS SSD volumes」(Amazon EBS ドキュメント)
- 「EC2 インスタンスの SSD インスタンスストアボリューム」 (Amazon EC2 ドキュメント)
- 「Amazon CloudWatch metrics for Amazon EBS」(Amazon EBS ドキュメント)
- 「Specifications for Amazon EC2 storage optimized instances」(Amazon EC2 ドキュメント)
- Amazon FSx for NetApp ONTAP で NetApp SnapCenter を使用して SQL Server ワークロードを保護する (AWS ストレージブログ)
- Amazon EC2 に関するよくある質問 (AWS 製品ページ)

Compute Optimizer を使用して SQL Server ライセンスを最適化する

を使用して SQL Server のライセンスを最適化する方法に関するガイダンス AWS Compute Optimizer。

概要

AWS Compute Optimizer は、Amazon Elastic Compute Cloud (Amazon EC2) 上の Microsoft SQL Server ワークロードのライセンス最適化の機会を推奨できます。Compute Optimizer は、ライセンスコストを削減するための自動推奨事項を提供できます。Compute Optimizer からの推奨事項は、Microsoft SQL Server ライセンスを持つ各 EC2 インスタンスの横に表示されます。提供される情報には、推奨される節約の機会、EC2 インスタンスのオンデマンド料金、時間単位のライセンス持ち込み (BYOL) 料金が含まれます。この情報は、ライセンスエディションをダウングレードすべきかどうかを判断するのに役立ちます。

Compute Optimizer は、推論されるワークロードタイプによって Amazon EC2 上の SQL Server インスタンスを自動的に検出します。ライセンスの推奨事項を表示するには、Compute Optimizer で SQL Server インスタンスを選択し、読み取り専用データベース認証情報を使用して Amazon

CloudWatch Application Insights で認証します。Compute Optimizer は、SQL Server Enterprise Edition の機能を使用しているかどうかを分析します。Enterprise Edition の機能が使用されていない場合、Compute Optimizer は、ライセンスコストを削減するために Standard Edition にダウングレードすることを推奨します。

Compute Optimizer を使用して、SQL Server ワークロードを実行する Amazon EC2 インスタンスのサイズ設定に関する推奨事項を示すこともできます。詳細については、このガイドの「Compute Optimizer を使用して SQL Server のサイズ設定を最適化する」を参照してください。

コスト最適化の推奨事項

Compute Optimizer のライセンス推奨事項は、Microsoft SQL Server で使用している機能を評価し、ワークロードに対して最もコスト効率が高いエディションを選択するのに役立ちます。SQL Server Enterprise Edition は、Standard Edition よりかなり高額です。詳細については、このガイドの「SQL Server エディションの比較」および Microsoft ウェブサイトの「SQL Server 2022 pricing」を参照してください。時間をかけて、SQL Server フリートを評価して推奨事項を提供するように Compute Optimizer を設定すれば、ライセンスコストを大幅に削減できます。

[ライセンスの詳細] ページには、次の情報が表示されます。

- この表を使用して、現在のライセンス設定 (エディション、モデル、インスタンスコア数など) を Compute Optimizer の推奨事項と比較します。
- 使用率グラフを使用して、分析期間中に使用された Enterprise Edition の機能の数を確認します。

詳細については、Compute Optimizer ドキュメントの「Viewing details of a commercial software license recommendation」を参照してください。

Compute Optimizer を設定する

Compute Optimizer は、`mssql_enterprise_features_used` メトリクスを使用して商用ソフトウェアライセンスを分析します。このメトリクスの詳細については、「Metrics for commercial software licenses」を参照してください。

1. Compute Optimizer にオプトインするための適切なアクセス許可があることを確認します。詳細については次を参照してください:
   - Compute Optimizer にオプトインするポリシー
   - スタンドアロン AWS アカウントに Compute Optimizer へのアクセス権を付与するポリシー
   - 組織の管理アカウントに Compute Optimizer へのアクセスを許可するポリシー

2. CloudWatch アプリケーションインサイトに必要なインスタンスロールとポリシーをアタッチします。手順については、「商用ソフトウェアライセンス推奨を有効にするポリシー」を参照してください。
3. Microsoft SQL Server データベースの認証情報を使用して CloudWatch アプリケーションインサイトを有効にします。手順については、CloudWatch ドキュメントの「AWS Management Console を使用してモニタリングするようにアプリケーションを設定する」を参照してください。

> **Note**
>
> 商用ソフトウェアライセンスの推奨事項を生成するには、連続して 30 時間以上の CloudWatch のメトリクスデータが必要です。詳細については、「CloudWatch metric requirements」を参照してください。

4. 次の SQL クエリを使用して、CloudWatch Application Insights の最小特権アクセスを設定します。

```
GRANT VIEW SERVER STATE TO [LOGIN];
GRANT VIEW ANY DEFINITION TO [LOGIN];
```

これにより、新しいサービス、PrometheusSqlExporterSQL が有効になります。

5. ターゲット AWS アカウント または組織管理アカウントから、Compute Optimizer にオプトインします。手順については、「アカウントにオプトインする」を参照してください。

> **Note**
>
> オプトイン後の検出結果と最適化のレコメンデーションの生成には、最大 24 時間かかることがあります。

6. Compute Optimizer コンソールのナビゲーションペインで、[ライセンス] を選択します。
7. [検出結果] 列で、検出結果が [メトリクスが不十分] であるインスタンスを検索します。Compute Optimizer は、CloudWatch Application Insights が有効になっていない、または権限が不十分であることを検出した場合、この検出結果を返します。詳細については、「Finding reasons」を参照してください。これらの検出結果を解決するには、以下を実行します。

   a. インスタンスを選択します。

   b. シークレットを追加します。

c. インスタンスロールとポリシーがアタッチされていることを確認します。

d. [ライセンスのレコメンデーションを有効にする] を選択します。

8. [検出結果] 列で、検出結果が [最適化されていません] であるインスタンスを検索します。Compute Optimizer は、お客様の Amazon EC2 インフラストラクチャが支払い対象の Microsoft SQL Server ライセンス機能をまったく使用していないことを検出した場合、この検出結果を返します。詳細については、「Finding reasons」を参照してください。これらの検出結果を解決するには、以下を実行します。

   a. インスタンスを選択します。

   b. 現在のライセンスエディションと推奨エディションを比較します。

   c. 現在のライセンス使用率グラフを確認します。

   d. ライセンスをダウングレードする場合は、[推奨事項の導入] を選択します。

   e. 要件を確認し、手順に従ってライセンスをダウングレードします。プロセスを自動化する場合は、「ドキュメントを使用した AWS Systems Manager SQL Server Enterprise Edition のダウングレード」を参照してコストを削減します (AWS ブログ) 。

その他のリソース

- による Microsoft SQL Server ライセンスコストの削減 AWS Compute Optimizer (AWS ブログ)
- とは AWS Compute Optimizer (AWS ドキュメント)
- 「Viewing commercial software license recommendations」(AWS ドキュメント)
- 「Downgrade your Microsoft SQL Server edition」(AWS ドキュメント)
- 「Microsoft SQL Server on AWS」(AWS)
- 「AWSでの Microsoft ライセンシング」(AWS)
- 「Microsoft SQL Server 2019 Pricing」(Microsoft)
- 「Microsoft SQL Server 2022 Pricing」(Microsoft)

Compute Optimizer を使用して SQL Server のサイズ設定を最適化する

概要

AWS Compute Optimizer は、データベース管理者 (DBA) が Amazon Elastic Compute Cloud (Amazon EC2) で Microsoft SQL Server ワークロードを検出し、EC2 インスタンスのサイズを適正化してライセンスコストを最大 25% 削減するのに役立ちます。Compute Optimizer の推定ワークロードタイプ機能は、機械学習 (ML) を使用し、 AWS リソースで実行されている可能性のあるアプリケーションを自動的に検出します。Compute Optimizer には、推論されるワークロードタイプとしての SQL Server のサポートが含まれています。推論されるワークロードタイプ機能を使用すると、Amazon EC2 インスタンスで実行されている特定のワークロードに基づいてコスト削減の機会を正確に特定できます。

この機能により、SQL Server など、サポートされている推論されるワークロードタイプ別にコスト削減の機会を分類できます。Compute Optimizer は、過剰にプロビジョニングされた SQL Server EC2 インスタンスを自動的に検出できます。EC2 コンソールに切り替えてインスタンスをダウンサイズすることで、ライセンスとインフラストラクチャのコストを削減できます。

Compute Optimizer を使用して、SQL Server ライセンスの推奨事項を示すこともできます。詳細については、このガイドの「Compute Optimizer を使用して SQL Server ライセンスを最適化する」を参照してください。

Compute Optimizer を設定する

SQL Server の推定ワークロードで Compute Optimizer を使用する手順については、「パフォーマンスの最適化とライセンスコストの削減: Amazon EC2 SQL Server インスタンス AWS Compute Optimizer の活用」(AWS ブログ) を参照してください。スタンドアロンアカウント、組織のメンバーであるアカウント、組織の管理アカウントに対してオプトインできます。スタンドアロンアカウントとメンバーアカウントの場合は、オプトインすると、そのアカウントに対してのみ Compute Optimizer が有効になります。組織の管理アカウントの場合は、そのアカウントでのみ Compute Optimizer を有効にするか、組織内のすべてのメンバーアカウントに対して有効にするかを選択できます。

Compute Optimizer オプトインプロセスでは、 AWS Identity and Access Management (IAM) サービスにリンクされたロールが自動的に作成されます。詳細については、「AWS Compute Optimizerのサービスにリンクされたロールの使用」を参照してください。

Compute Optimizer は、CPU、I/O、ネットワーク、Amazon Elastic Block Store (Amazon EBS) の使用状況などの Amazon CloudWatch メトリクスに基づいてリソースを分析します。推奨事項を生成するには、過去 14 日間における連続した 30 時間以上の CloudWatch メトリクスデータが必要です。拡張インフラストラクチャメトリクス機能を有効にすると、使用率メトリクスが 93 日に延長されます。詳細については、Compute Optimizer ドキュメントの「CloudWatch metric requirements」と「Enhanced infrastructure metrics」を参照してください。

Compute Optimizer は、vCPU、メモリ、ストレージ、ネットワーク、リスク、移行の労力に基づいて、オプションと各オプションに関連する節約を提供します。CloudWatch メトリクスダッシュボードを使用して、推奨事項の作成に使用されているデータを分析できます。このデータを使用すると、SQL Server ワークロードを実行している EC2 インスタンスのサイズを適正化できます。インスタンスタイプの変更方法の詳細については、Amazon EC2 ドキュメントの「Amazon EC2 インスタンスタイプの変更」を参照してください。

その他のリソース

- AWS Compute Optimizer Microsoft SQL Server ワークロードを識別してフィルタリングする (AWS)
- パフォーマンスの最適化とライセンスコストの削減: Amazon EC2 SQL Server インスタンス AWS Compute Optimizer の活用 (AWS ブログ)
- とは AWS Compute Optimizer (AWS ドキュメント)
- EC2 インスタンスのレコメンデーションの表示 (AWS ドキュメント)

SQL Server ワークロードの Trusted Advisor レコメンデーションを確認する

概要

AWS Trusted Advisor は、 AWS ベストプラクティスに従うのに役立つ推奨事項を提供します。は、使用状況、設定、支出を分析することで、コスト削減、システムの可用性とパフォーマンスの向上、セキュリティギャップの解消に役立つ実用的な推奨事項 Trusted Advisor を提供します。このセクションでは、 での SQL Server ワークロードの運用コストを削減するのに役立つ Trusted Advisor チェックに焦点を当てます AWS クラウド。

コスト最適化の推奨事項

Trusted Advisor は、Amazon Elastic Compute Cloud (Amazon EC2) で SQL Server ワークロードを最適化するのに役立つ推奨事項を提供します。このチェックでは、SQL Server ワークロードを検査し、最適化が必要なインスタンスを自動的に一覧表示します。 Trusted Advisor レコメンデーションを運用することで、コストを削減し、組織のセキュリティ体制を向上させることができます。

Microsoft SQL Server に焦点を当てた Trusted Advisor チェックを次に示します。

- 過剰にプロビジョニングされた Amazon EC2 インスタンス (Microsoft SQL Server 向け) – このチェックでは、SQL Server を実行している Amazon EC2 インスタンスを分析し、インスタンスが SQL Server ソフトウェアの vCPU 制限を超えた場合に警告します。例えば、SQL Server Standard Edition を使用するインスタンスでは、最大 48 個の vCPU を使用できます。SQL Server Web を使用するインスタンスでは、最大 32 個の vCPU を使用できます。

Edition	vCPU 最小	vCPU 最大
Web	4	32
標準	4	48
Enterprise	4	OS の上限

- Amazon EC2 インスタンスの統合 (Microsoft SQL Server 向け) – このチェックでは、Amazon EC2 インスタンスを分析し、インスタンスが SQL Server ライセンスの最小数より少ない場合に警告します。小さめの SQL Server インスタンスを統合して、コストの削減に役立てることができます。ライセンス込みの小さな SQL Server インスタンスが多数ある場合は、統合することを検討してください。「Microsoft SQL Server 2019 licensing guide」によると、SQL Server ではインスタンスごとに最低 4 つの vCPU ライセンスが必要です。これらのデータベースを統合すると、ライセンスコストを節約できます。インスタンス上のデータベースの数、データベースの最大サイズ、データベースの合計サイズに基づいて決定できます。統合は、SQL Server の Web、Standard、Enterprise の各エディションでサポートされています。詳細については、「Consolidating SQL Server Databases」(Microsoft ブログ記事) を参照してください。

  AWS では、大規模な本番稼働用データベースを 1 つのサーバーにのみ配置することはお勧めしません。ただし、開発、テスト、ステージングなど、非本番環境に使用される小規模なデータベースを統合することは可能です。これは現在の SQL Server の使用状況によって異なります。使用量の少ないデータベースがある場合は、1 つのサーバーに統合できます。

設定 Trusted Advisor

SQL Server に焦点を当てたチェックを評価するには、以下を実行します Trusted Advisor。

1. AWS マネジメントコンソールにサインインします。
2. AWS Trusted Advisor コンソールを開きます。
3. ナビゲーションペインの [推奨事項] の下で、[コスト最適化] を選択します。
4. [コスト最適化チェック] リストで、[Amazon EC2 インスタンスの統合 (Microsoft SQL Server 向け)] のチェックと [過剰にプロビジョニングされた Amazon EC2 インスタンス (Microsoft SQL Server 向け)] のチェックのステータスを確認します。
   - 緑色のチェック記号は、Amazon EC2 インスタンスが最適に設定されていることを示します。
   - オレンジ色のアラート記号は、改善の機会があることを示します。
5. チェックを選択すると、詳細と推奨事項が表示されます。
6. SQL Server ワークロードを実行している Amazon EC2 インスタンスを最適化するには、チェックで提供された指示に従います。
7. インスタンスを定期的にモニタリングし、チェックを定期的に更新します。

その他のリソース

- Trusted Advisor チェックリファレンス (AWS ドキュメント)
- 「Microsoft SQL Server on AWS」(AWS)
- 「AWSでの Microsoft ライセンシング」(AWS)
- 「SQL Server 2019 pricing」(Microsoft)
- AWS Launch Wizard for SQL Server (AWS ドキュメント)

コンテナ

モダナイズは、モノリスをマイクロサービスに分解する、サーバーレス関数 (AWS Lambda) を使用してイベントドリブンにアプリケーションを再設計する、SQL Server から Amazon Aurora または専用マネージドデータベースにデータベースを再利用するなど、多くのオプションを提供するトランスフォーメーションジャーニーです。.NET Framework アプリケーションを Linux および Windows コンテナにリプラットフォームするためのモダナイズパスウェイは、他のモダナイズオプションよりも労力が少なくて済みます。コンテナには次の利点があります。

- イノベーションを加速する – コンテナに移行すると、アプリケーションの構築、テスト、デプロイを含む開発ライフサイクルのステージの自動化が容易になります。これらのプロセスを自動化することで、開発チームと運用チームはイノベーションに集中する時間が増えます。
- 総保有コスト (TCO) を削減する – コンテナに移行すると、ライセンス管理およびエンドポイント保護ツールへの依存度も低下します。コンテナはエフェメラル単位のコンピューティングであるため、パッチ適用、スケーリング、バックアップ、復元などの管理タスクを自動化および簡素化できます。これにより、コンテナベースのワークロードの管理と運用の TCO が削減されます。最後に、コンテナは仮想マシンと比較してより効率的です。これは、コンテナを使用して、より良い分離を提供することでアプリケーションの配置を最大化できるためです。これにより、アプリケーションのインフラストラクチャリソースの使用率が向上します。
- リソース使用率が向上する – コンテナを使用してアプリケーションの配置を最大化できるため、コンテナは仮想マシンと比較してより効率的です。これにより、より良い分離を提供することでアプリケーションのインフラストラクチャリソースの使用率が向上します。
- スキルギャップを埋める - イマージョンデー AWS を提供して、コンテナテクノロジーと DevOps プラクティスの開発チームをスキルアップします。

このセクションでは、次のトピックについて説明します。

- Windows アプリケーションをコンテナに移行する
- Amazon ECS の AWS Fargate タスクのコストを最適化する
- Amazon EKS のコストを可視化する
- Windows アプリケーションを App2Container でリプラットフォームする

ライセンス情報については、「AWS と Microsoft に関するよくある質問」の「ライセンシング」セクションを参照するか、microsoft@amazon.com までメールでお問い合わせください。

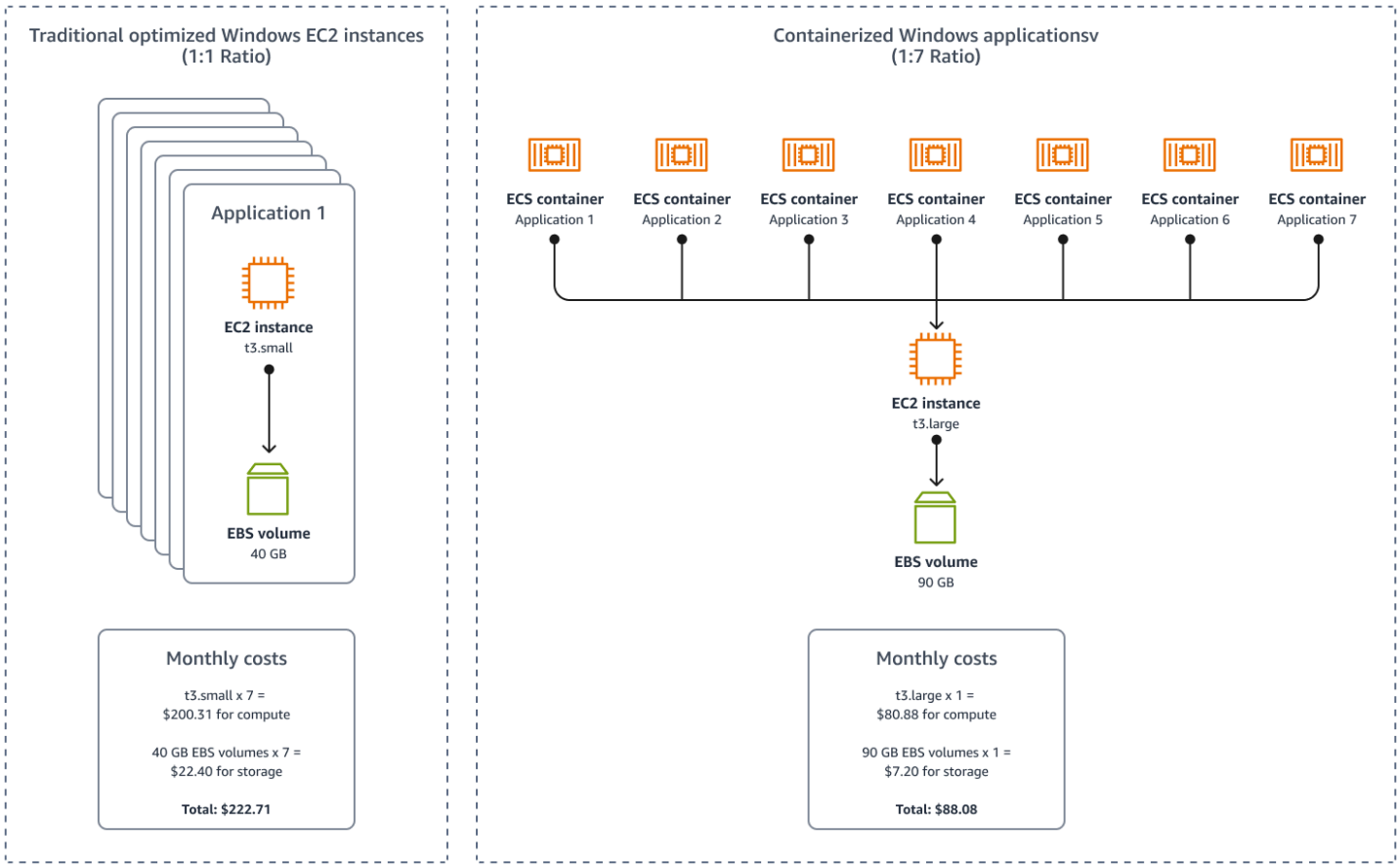
Windows アプリケーションをコンテナに移行する

概要:

「CNCF Annual Survey 2021」によると、組織の 96% が、インフラストラクチャをモダナイズするためにコンテナを使用または評価しています。これは、コンテナが組織のリスクを軽減し、運用効率とスピードを向上させ、俊敏性を実現するのに役立つためです。コンテナを使用して、アプリケーションの実行コストを削減することもできます。このセクションでは、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、 など、 AWS コンテナサービス全体でコスト効率の高いコンテナの実行に関する推奨事項を提供しますAWS Fargate。

コスト上の利点

次のインフォグラフィックは、AWS 最適化とライセンス評価 (AWS OLA) の推奨事項に基づいて ASP.NET Framework アプリケーションを Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに統合することで、企業が達成できるコスト削減を示しています。次のインフォグラフィックは、アプリケーションを Windows コンテナに移行することで達成できる追加の削減額を示しています。



AWS OLA は、ビジネスが個々の t3.small インスタンスにリフトアンドシフトすることを推奨しました。次のパフォーマンス使用率分析が示すように、企業は、オンプレミスサーバーで 7 つの ASP.NET アプリケーションを実行することで、この削減を実現できます。

	Server name	Storage	Operating system	On-premises CPU AVG utilization	On-premises CPU peak utilization	On-premises RAM (GB)	On-premises RAM AVG utilization (GB)	On-premises RAM peak utilization (GB)	Instance size	vCPU	RAM (GB)
1	AppServer01	60	Windows Server 2012	7.00%	17.00%	8	13.50%	17.10%	t3.small	2	2
2	AppServer02	39	Windows Server 2012	20.07%	22.00%	16	7.50%	12.40%	t3.small	2	2
3	AppServer03	39	Windows Server 2012	24.00%	25.50%	16	8.80%	11.90%	t3.small	2	2
4	AppServer04	4	Windows Server 2012	21.40%	24.00%	16	7.80%	10.70%	t3.small	2	2
5	AppServer05	40	Windows Server 2012	21.30%	23.00%	16	8.20%	12.00%	t3.small	2	2
6	AppServer06	39	Windows Server 2012	21.50%	23.50%	16	7.90%	10.90%	t3.small	2	2
7	AppServer07	39	Windows Server 2012	21.60%	22.90%	16	8.40%	11.50%	t3.small	2	2

さらなる分析により、企業はコンテナでワークロードを実行することでコストをさらに削減できることが明らかになりました。コンテナは、CPU、RAM、ディスク使用量などのシステムリソースに対するオペレーティングシステムのオーバーヘッドを削減します (次のセクションで説明)。このシナリオでは、企業は 7 つのアプリケーションすべてを 1 つの t3.large インスタンスに統合し、さらに 3 GB の RAM の空きを確保しています。コンテナに移行すると、Amazon EC2 の代わりにコンテナを使用することで、コンピューティングおよびストレージ全体で平均 64% のコスト削減を実現できます。

コスト最適化の推奨事項

次のセクションでは、アプリケーションを統合してコンテナを使用することでコストを最適化するための推奨事項を示します。

Amazon EC2 での Windows のフットプリントの削減

Windows コンテナを使用すると、より多くのアプリケーションをより少ない EC2 インスタンスに統合できるため、Amazon EC2 での Windows のフットプリントを削減できます。例えば、ASP.NET アプリケーションが 500 個あるとします。Amazon EC2 で Windows のアプリケーションごとに 1 つのコアを実行している場合、これは 500 個の Windows インスタンス (t3.small) に相当します。Windows コンテナ (t3.large を使用) の使用に 1:7 の比率 (EC2 インスタンスのタイプ/サイズに応じて大幅に増加する可能性あり) を想定した場合、必要な Windows インスタンスは約 71 個のみです。これは、Amazon EC2 での Windows のフットプリントが 85.8% 減少することを意味します。

Windows ライセンスコストの削減

Windows インスタンスをライセンスする場合、そのインスタンスで実行されているコンテナをライセンスする必要はありません。その結果、Windows コンテナを使用して ASP.NET アプリケーションを統合することで、Windows ライセンスコストを大幅に削減できます。

ストレージフットプリントの削減

新しい EC2 インスタンスを起動するたびに、オペレーティングシステムを格納するための新しい Amazon Elastic Block Store (Amazon EBS) ボリュームを作成し、その料金を支払います。これがスケールすると、それに応じてコストもスケールします。コンテナを使用すると、すべてのコンテナが同じ基本オペレーティングシステムを共有するため、ストレージコストを削減できます。さらに、コンテナはレイヤーの概念を使用して、コンテナイメージのイミュータブルな部分を、そのイメージに基づいて実行中のすべてのコンテナに再利用します。前述のシナリオ例では、すべてのコンテナが .NET Framework を実行しているため、すべてが中間およびイミュータブルな ASP.NET フレームワークレイヤーを共有しています。

サポート終了サーバーのコンテナへの移行

Windows Server 2012 および Windows Server 2012 R2 のサポートは、2023 年 10 月 10 日に終了しました。Windows Server 2012 以前のバージョンで実行されているアプリケーションは、新しいオペレーティングシステムで実行するようにコンテナ化することで移行できます。これにより、コンテナが提供するコスト効率、リスクの低減、運用効率、速度、俊敏性を活用しながら、非準拠のオペレーティングシステムでアプリケーションを実行することが回避されます。

このアプローチで考慮すべき注意点は、アプリケーションが、現在使用中のオペレーティングシステムのバージョンに関連する特定の API (COM Interop など) を必要とする場合です。この場合は、アプリケーションを新しい Windows バージョンに移行することをテストする必要があります。Windows コンテナは、ベースコンテナイメージ (Windows Server 2019 など) を、コンテナホストのオペレーティングシステム (Windows Server 2019 など) に合わせます。テストしてコンテナに移行すると、Dockerfile 内のベースイメージを変更し、最新バージョンの Windows を実行している新しいホストのセットにデプロイすることで、将来のオペレーティングシステムのアップグレードが容易になります。

サードパーティー製の管理ツールとライセンスの削除

サーバーフリートを管理するには、パッチ適用と設定管理に複数のサードパーティー製のシステムオペレーションツールを使用する必要があります。これにより、インフラストラクチャ管理が複雑になり、サードパーティーのライセンスコストが発生することがよくあります。でコンテナを使用する場

合 AWS、オペレーティングシステム側で何も管理する必要はありません。コンテナランタイムはコンテナを管理します。つまり、基盤となるホストはエフェメラルであり、簡単に置き換えることができます。コンテナホストを直接管理しなくても、コンテナを実行できます。さらに、 などの無料のツールを使用して、ホスト AWS Systems Manager Session Manager に簡単にアクセスし、問題をトラブルシューティングできます。

コントロールと移植性の向上

コンテナを使用すると、EC2 インスタンスよりも CPU や RAM などのサーバーリソースをより細かく制御できます。EC2 インスタンスの場合、インスタンスファミリー、インスタンスタイプ、CPU オプションを選択して CPU と RAM を制御できます。ただし、コンテナでは、ECS タスク定義のコンテナまたは Amazon EKS のポッドに割り当てる CPU または RAM の量を正確に定義できます。実際のところ、Windows コンテナでは、コンテナレベルの CPU とメモリを指定することをお勧めします。このレベルの粒度はコスト上の利点をもたらします。次のサンプルコードについて検討します。

```
json
{
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/demo-
service:1",
    "containerDefinitions": [
        {
            "name": "demo-service",
            "image": "mcr.microsoft.com/dotnet/framework/samples:aspnetapp-
windowsservercore-ltsc2019",
            "cpu": 512,
            "memory": 512,
            "links": [],
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
```

イノベーションの加速

コンテナに移行すると、アプリケーションの構築、テスト、デプロイを含む開発ライフサイクルのステージの自動化が容易になります。これらのプロセスを自動化すると、開発チームと運用チームがイノベーションに集中する時間が増えます。

TCO の削減

コンテナに移行すると、ライセンス管理ツールやエンドポイント保護ツールへの依存が減ることがよくあります。コンテナはエフェメラル単位のコンピューティングであるため、パッチ適用、スケーリング、バックアップ、復元などの管理タスクを自動化および簡素化できます。これにより、コンテナベースのワークロードの管理と運用の TCO を削減できます。コンテナは、アプリケーションのインフラストラクチャリソースの使用率を高めることができるように、アプリケーションの配置を最大化できるため、仮想マシンと比較してより効率的です。

スキルギャップの解消

AWS は、コンテナと DevOps テクノロジーに関するカスタマー開発チームをスキルアップするためのプログラムとイマージョンデーを提供します。これには、実践的なコンサルティングと支援が含まれます。

.NET 5+ へのリファクタリングと Linux コンテナの使用

.NET Framework アプリケーションをコンテナに移行することでコストを削減できますが、レガシー .NET アプリケーションを AWSのクラウドネイティブな代替手段にリファクタリングすると、さらにコスト削減を実現できます。

ライセンスコストの削除

Windows 上の .NET Framework から Linux 上の .NET Core にアプリケーションをリファクタリングすると、約 45% のコスト削減につながります。

最新の拡張機能へのアクセス

Windows 上の .NET Framework から Linux 上の .NET Core にアプリケーションをリファクタリングすると、Graviton2 などの最新の拡張機能にアクセスできます。Graviton2 は、同等のインスタンスよりもパフォーマンスに対して 40% 有利な価格を提供します。

セキュリティとパフォーマンスの改善

Windows 上の .NET Framework から Linux 上の .NET Core コンテナにアプリケーションをリファクタリングすると、セキュリティとパフォーマンスが向上します。これは、最新のセキュリティパッチを取得し、コンテナの分離によるメリットを得て、新機能にアクセスできるためです。

IIS の 1 つのインスタンスで多くのアプリケーションを実行する代わりに Windows コンテナを使用する

インターネットインフォメーションサービス (IIS) を使用して 1 つの EC2 Windows インスタンスで複数のアプリケーションを実行する代わりに Windows コンテナを使用することによる次の利点を考慮します。

- セキュリティ – コンテナは、IIS レベルでの分離では達成されないセキュリティレベルを追加設定なしで提供します。1 つの IIS ウェブサイトまたはアプリケーションが侵害された場合、他のすべてのホストサイトが公開され、脆弱になります。コンテナエスケープはまれであり、ウェブ脆弱性を通じてサーバーを制御するよりも悪用するのが難しい脆弱性です。
- 柔軟性 – コンテナをプロセス分離で実行し、独自のインスタンスを持つ機能により、より詳細なネットワークオプションが可能になります。コンテナは、多くの EC2 インスタンスにわたる複雑な分散方法も提供します。アプリケーションを 1 つの IIS インスタンスに統合しても、これらの利点はありません。
- 管理オーバーヘッド – サーバー名表示 (SNI) は、管理と自動化を必要とするオーバーヘッドを作成します。また、パッチ適用、BSOD のトラブルシューティング (自動スケーリングが有効になっていない場合)、エンドポイントの保護など、一般的なオペレーティングシステム管理オペレーションに取り組む必要があります。セキュリティのベストプラクティスに従って IIS サイトを設定することは、時間のかかる継続的なアクティビティです。場合によっては、信頼レベルを設定する必要があります。これにより、管理オーバーヘッドも増加します。コンテナは、ステートレスでイミュータブルになるように設計されています。最後に、代わりに Windows コンテナを使用すると、デプロイはより高速で、より安全で、繰り返し可能なものになります。

次の手順

レガシーワークロードを実行するために最新のインフラストラクチャに投資すると、組織に大きなメリットがあります。 AWS コンテナサービスを使用すると、オンプレミスでもクラウドでも基盤となるインフラストラクチャを簡単に管理できるため、イノベーションとビジネスニーズに集中できます。クラウド内のすべてのコンテナのほぼ 80% が AWS 今日稼働しています。 は、ほぼすべてのユースケースに対して豊富なコンテナサービス AWS を提供します。開始するには、「AWSのコンテナ」を参照してください。

その他のリソース

- ECS キャパシティプロバイダーと EC2 スポットインスタンスを使用してコンテナワークロードのコストを最適化する (AWS ブログ)

- 「Cost Optimization Checklist for Amazon ECS and AWS Fargate」(AWS ブログ)
- Amazon EKS on AWS Graviton2 の一般提供: マルチアーキテクチャアプリに関する考慮事項 (AWS ブログ)
- での Kubernetes のコスト最適化 AWS (AWS ブログ)
- Karpenter 統合による Kubernetes コンピューティングコストの最適化 (AWS ブログ)

Amazon ECS の AWS Fargate タスクのコストを最適化する

概要:

適切なサイジング AWS Fargate タスクは、コスト最適化の重要なステップです。多くの場合、アプリケーションは Fargate タスクに対して任意のサイズ設定で構築され、再検討されることはありません。これにより、Fargate タスクの過剰プロビジョニングや不要な支出が発生する可能性があります。このセクションでは、AWS Compute Optimizer を使用して実用的な推奨事項を提供し、Fargate で実行されている Amazon Elastic Container Service (Amazon ECS) サービスのタスク CPU およびメモリを最適化する方法について説明します。Compute Optimizer は、これらの推奨事項を採用した場合のコストへの影響も定量化します。これにより、コスト削減の機会のサイズに基づいて最適化作業に優先順位を付けることができます。Compute Optimizer 推奨事項は、タスクをダウンサイジングするためのコンテナレベルの CPU とメモリの設定を提供します。

コスト上の利点

Fargate で Amazon ECS タスクのサイズを適切に設定すると、長時間実行されるタスクのコストを 30～70% 削減できます。アプリケーションパフォーマンスメトリクスを確認してタスクサイズを適切にサイズ設定しなくても、EC2 コンピューティングインスタンスで使用されているのと同じ考え方をコンテナのサイズ設定に適用できます。これにより、オーバーサイズの Fargate タスクが発生し、アイドル状態のリソースのコストが増加します。Compute Optimizer を使用して、適切なサイズ設定の機会を事後対応的に明らかにすることができます。理想的には、アプリケーション所有者は、特定のアプリケーションパフォーマンスメトリクスを確認し、オペレーティングシステムのオーバーヘッドを削除して、適切なタスクサイズが指定されていることを確認します。詳細については、このガイドの「Windows アプリケーションをコンテナに移行する」セクションを参照してください。

コスト最適化の推奨事項

このセクションでは、Compute Optimizer を使用して Fargate タスクで Amazon ECS のサイズを適正化するための推奨事項を示します。

コスト最適化プロセスの一環として、以下を実行することをお勧めします。

- Compute Optimizer を有効にする
- Compute Optimizer の結果を使用する
- 適切なサイズになるようにタスクにタグを付ける
- AWS 請求ツールを操作するためにコスト配分タグを有効にする
- 適切なサイズ設定の推奨事項を実装する
- Cost Explorer で前と後のコストを確認する

Compute Optimizer を有効にする

組織レベルまたは AWS Organizationsの単一アカウントレベルで AWS Compute Optimizer を有効にできます。組織全体の設定では、すべてのメンバーアカウントのフリート全体の新規および既存のインスタンスの継続的なレポートが提供されます。これにより、適切なサイズ設定を、ポイントインタイムアクティビティではなく、繰り返し行うアクティビティにすることができます。

組織レベル

ほとんどの組織にとって、Compute Optimizer を使用する最も効率的な方法は組織レベルです。これにより、マルチアカウントとマルチリージョンの組織への可視性が提供され、レビューのためにデータを 1 つのソースに一元化できます。組織レベルでこれを有効にするには、以下を実行します。

1. 必要なアクセス許可を持つロールを使用してAWS Organizations 管理アカウントにサインインし、この組織内のすべてのアカウントにオプトインすることを選択します。組織で、すべての機能が有効になっている必要があります。
2. 管理アカウントを有効にしたら、アカウントにサインインし、他のすべてのメンバーアカウントを表示して、その推奨事項を参照できます。

> **Note**
>
> Compute Optimizer の委任管理者アカウントを設定するのがベストプラクティスです。これにより、最小特権の原則を実行し、組織全体のサービスへのアクセスを提供しながら、AWS Organizations 管理アカウントへのアクセスを最小限に抑えることができます。

単一アカウントレベル

コストの高いアカウントをターゲットにしているが、 AWS Organizationsにアクセスできない場合は、そのアカウントとリージョンに対して Compute Optimizer を有効にできます。オプトインプロセスの詳細については、「 の開始方法 AWS Compute Optimizer」を参照してください。

> **Note**
>
> レコメンデーションは毎日更新され、生成までに最大 12 時間かかることがあります。Compute Optimizer が Fargate での Amazon ECS の推奨事項を生成するには、過去 14 日間に 24 時間分のメトリクスを必要とすることに注意してください。詳細については、Compute Optimizer ドキュメントの「Requirements for Amazon ECS services on Fargate」を参照してください。

Compute Optimizer は、Fargate 上の Amazon ECS サービスの、次の Amazon CloudWatch と Amazon ECS の使用メトリクスを自動的に分析します。

- `CPUUtilization` – サービスで使用されている CPU キャパシティのパーセンテージ。
- `MemoryUtilization` – サービスで利用されるメモリの割合。

Compute Optimizer の結果を使用する

1 つのアカウントと 1 つのリージョン内で適切なサイズ変更を行うことに焦点を当てた例について考えてみましょう。この例では、Compute Optimizer はすべてのアカウントで組織レベルで有効になっています。適切なサイズ設定は破壊的なプロセスであり、ほとんどの場合、数週間にわたるスケジュールされたメンテナンス期間中にアプリケーション所有者が正確に実行するプロセスであるという点に注意してください。

組織の管理アカウント内から Compute Optimizer に移動する場合は (次の手順を参照)、調査するアカウントを選択できます。この例では、1 つのタスクが、`us-east-1` で過剰にプロビジョニングされている 1 つのアカウントで実行されています。焦点は、Amazon ECS サービスの推奨サイズへのサイズ変更です。

1. Compute Optimizer コンソールを開きます。
2. [ダッシュボード] ページで、Fargate のすべての Amazon ECS サービスを表示するには、[検出結果] = [過剰なプロビジョニング] でフィルタリングします。

3. Fargate で過剰にプロビジョニングされた ECS サービスの詳細な推奨事項を確認するには、下にスクロールして [推奨事項を表示] を選択します。
4. [エクスポート] を選択し、後で使用するためにファイルを保存します。

> **Note**
>
> 今後のレビューのために推奨事項を保存するには、Compute Optimizer が各リージョンで書き込める S3 バケットが必要です。詳細については、Compute Optimizer ドキュメントの「Amazon S3 bucket policy for AWS Compute Optimizer」を参照してください。

Compute Optimizer からの推奨事項を表示するには、以下を実行します。

1. Compute Optimizer コンソールで、[エクスポートの推奨事項] ページに移動します。
2. [S3 バケットの宛先] で、S3 バケットを選択します。
3. [フィルタをエクスポート] セクションの [リソースタイプ] で、[Fargate での ECS サービス] を選択します。
4. [Fargate での ECS サービスの推奨事項] ページで、Fargate の ECS サービスの 1 つをドリルダウンし、Compute Optimizer の CPU とメモリの推奨事項を確認します。例えば、「現在の設定と推奨タスクサイズの比較」セクションと「現在の設定と推奨コンテナサイズの比較」セクションの推奨事項を確認します。

適切なサイズが必要な Fargate の ECS サービスのリストを取得するには、以下を実行します。

1. Amazon S3 コンソールを開きます。
2. ナビゲーションペインで [バケット] を選択し、結果をエクスポートしたバケットを選択します。
3. [オブジェクト] タブで、オブジェクトを選択し、[ダウンロード] を選択します。
4. ダウンロードした結果で、検出結果列をフィルタリングして、Fargate の OVER_PROVISIONED Amazon ECS サービスのみを表示します。これは、適切なサイズ設定の対象となる予定の Amazon ECS サービスを示しています。
5. 後で使用するために、タスク定義をテキストエディタで保存します。

タグタスクの適切なサイズ設定

ワークロードにタグを付けることは、 AWSでリソースを整理するための強力なツールです。タグを使用すると、コストをきめ細かく可視化し、チャージバックを有効にできます。チャージバックと

自動化を処理するために AWS リソースにタグを追加する方法と戦略は多数あります。詳細については、「 リソースのタグ付けに関する AWS ホワイトペーパーのベストプラクティス」を参照してください。 AWS次の例では、AWS CloudShell を使用して、ターゲットアカウントおよび AWS リージョン内の Amazon ECS サービスの一部であるすべてのタスクにタグを付けます。

```
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="enabled"
# Get a list of ECS Clusters
ClustersArns=$( aws ecs list-clusters -query 'clusterArns' -output text)
for ClustersArn in $ClustersArns; do
 ServiceArns=$( aws ecs list-services -cluster $ClustersArn -query 'serviceArns' -
output text)
 for ServiceArn in $ServiceArns; do
  TasksArns=$( aws ecs list-tasks -cluster $ClustersArn -service-name $ServiceArn -
query 'taskArns' -output text)
  for TasksArn in $TasksArns; do
    aws ecs tag-resource -resource-arn $TasksArn -tags key=$TAG_KEY,value=$TAG_VALUE
  done
 done
done
```

次のコード例は、すべての Amazon ECS サービスへのタグ伝搬を有効にする方法を示しています。

```
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="enabled"
# Get a list of ECS Clusters
ClustersArns=$(aws ecs list-clusters --query 'clusterArns' --output text)
for ClustersArn in $ClustersArns; do
 ServiceArns=$(aws ecs list-services --cluster $ClustersArn --query 'serviceArns' --
output text)
 for ServiceArn in $ServiceArns; do
  aws ecs update-service --cluster $ClustersArn --service $ServiceArn --propagate-tags
 SERVICE &>/dev/null
  aws ecs tag-resource --resource-arn $ServiceArn --tags key=$TAG_KEY,value=$TAG_VALUE
 done
done
```

AWS 請求ツールを操作するためにコスト配分タグを有効にする

ユーザー定義のコスト配分タグをアクティブ化することをお勧めします。これにより、 AWS 請求ツール (AWS Cost Explorer や など AWS Cost and Usage Report) で Rightsizing タグが認識され、フィルタリングできるようになります。これを有効にしない場合、タグフィルタリングオプションとデータは使用できなくなります。コスト配分タグの使用の詳細については、 AWS Billing and Cost Management ドキュメントの「ユーザー定義のコスト配分タグのアクティブ化」を参照してください。

24 時間待ってから、次のセクションで適切なサイズ設定の推奨事項を実装する前に、Cost Explorer でタグを確認できます。これを行うには、Cost Explorer で [ライトサイジング] タグを検索します。

適切なサイズ設定の推奨事項を実装する

Compute Optimizer は、タスクまたはコンテナサイズの推奨事項を提供します。適切なサイズ設定の推奨事項を実装するには、以下を実行します。

1. Amazon ECS コンソールを開きます。
2. ナビゲーションバーから、タスク定義を含むリージョンを選択します。
3. ナビゲーションペインで、[Task Definitions] を選択します。
4. [Task definitions] (タスク定義) ページでタスクを選択し、[Create new revision] (新しいリビジョンを作成する) を選択します。
5. [Create new task definition revision (タスク定義の新しいリビジョンの作成) ページで変更を加えます。コンテナサイズの推奨事項を更新するには、[ECS タスク定義] の containerDefinitions ブロックで `cpu` と `memory` を更新します。例えば、次のようになります。

```
"containerDefinitions": [
 {
  "name": "your-container-name",
  "image": "your-image",
  "cpu": 1024,
  "memory": 2048,
 }
],
```

6. 情報を確認し、[Create] (作成) を選択します。

Amazon ECS サービスを更新するには、以下を実行します。

1. Amazon ECS コンソールを開きます。
2. [Clusters] (クラスター) ページで、クラスターを選択します。
3. [Cluster overview] (クラスターの概要ページ) で、サービスを選択し、[Update] (更新) を選択します。
4. [Task definition] (タスク定義) の場合、使用するタスク定義ファミリーとリビジョンを選択します。

高度な演算子の場合は、CloudShell を使用して Amazon ECS サービスを更新できます。例えば、次のようになります。

```
bash
#!/bin/bash
# Set variables
ClustersName="workshop-cluster"
ServiceName="lab7-fargate-service"
TaskDefinition="lab7-fargate-demo:3"
# update the service
aws ecs update-service --cluster $ClustersName --service $ServiceName --task-definition
 $TaskDefinition
```

前と後のコストを確認する

リソースのサイズを適切に設定したら、Cost Explorer を使用し、[ライトサイジング] タグを使用して、前と後のコストを表示できます。リソースタグを使用してコストを追跡できることを思い出してください。複数のタグレイヤーを使用することで、コストをきめ細かく可視化できます。このガイドで説明する例では、[ライトサイジング] タグを使用して、すべてのターゲットインスタンスに汎用タグを適用します。次に、[チーム] タグを使用して、リソースをさらに整理します。次のステップでは、アプリケーションタグを導入して、特定のアプリケーションを運用する際のコストへの影響をさらに示します。

単一のアカウントレベルで [ライトサイジング] タグを使用することで実現できるコスト削減の例を考えてみましょう。この例では、運用コストは 1 日あたり 30.26 USD から 7.56 USD になります。1 か月あたり 744 時間と仮定すると、適切なサイズ設定の前の年間コストは 11,044.9 USD です。適切なサイズ設定の後、年間コストは 2,759.4 USD に削減されます。これにより、このアカウントのコンピューティングコストが 75% 削減されます。これが大規模な組織全体に及ぼす影響を想像してみてください。

適切なサイズ設定ジャーニーを開始する前に、次の点を考慮してください。

- AWS には、コスト削減のための多くのオプションがあります。これには AWS OLA が含まれます。 は、移行する前にオンプレミスインスタンス AWS を確認します AWS。 AWS OLA には、適切なサイズの推奨事項とライセンスガイダンスも用意されています。
- Savings Plans を購入する前に、適切なサイズ設定をすべて完了してください。これにより、Savings Plans コミットメントでの過剰購入を回避できます。

次の手順

次のステップとして以下の手順の実行を推奨します。

1. 既存のランドスケープを確認し、Amazon EBS gp2 ボリュームを gp3 ボリュームに変換することを検討します。
2. Savings Plans を確認します。

その他のリソース

- Compute Optimizer の開始方法 (AWS ドキュメント)
- AWS リソースのタグ付けのベストプラクティス (AWS ホワイトペーパー)
- での Windows コンテナ AWS (AWS Workshop Studio)

Amazon EKS のコストを可視化する

概要:

Kubernetes デプロイのコストを効果的にモニタリングするには、全体像を把握する必要があります。唯一の固定かつ既知のコストは、Amazon Elastic Kubernetes Service (Amazon EKS) コントロールプレーンのコストです。これには、コンピューティングやストレージからネットワークまで、デプロイを構成する他のすべてのコンポーネントが含まれます。これは、アプリケーションのニーズに応じて変動する量です。

Kubecost を使用すると、名前空間やサービスから個々のポッドに至るまで Kubernetes インフラストラクチャのコストを分析し、そのデータをダッシュボードに表示できます。Kubecost は、コンピューティングやストレージなどのクラスター内コストと、Amazon Simple Storage Service (Amazon S3) バケットや Amazon Relational Database Service (Amazon RDS) インスタンスなどのクラスター外コストを明らかにします。Kubecost は、このデータに基づいて適切なサイズ設定の推奨を行い、システムに影響を与える可能性のある重要なアラートを表示します。Kubecost は AWS

Cost and Usage Report と統合して、Compute Savings Plans、リザーブドインスタンス、その他の割引プログラムによる節約額を表示できます。

コスト上の利点

Kubecost は、Amazon EKS デプロイのコストを視覚化するレポートとダッシュボードを提供します。これにより、クラスターから、コントローラー、サービス、ノード、ポッド、ボリュームなどのさまざまな各コンポーネントにドリルダウンできます。また、これにより、Amazon EKS 環境で実行されているアプリケーションの全体像を把握できます。この可視性を有効にすると、Kubecost の推奨事項に基づいて行動したり、各アプリケーションのコストをきめ細かく表示したりできます。Amazon EKS ノードグループのサイズを適切に設定すると、標準の EC2 インスタンスと同じ削減可能額が得られます。コンテナとノードのサイズを適切に設定できる場合は、コンテナの実行に必要なインスタンスのサイズと Auto Scaling グループに必要な EC2 インスタンスの数からコンピューティングの肥大化を排除できます。

コスト最適化の推奨事項

Kubecost を利用するには、以下を実行することをお勧めします。

1. 環境に Kubecost をデプロイする
2. Windows アプリケーションの詳細なコスト内訳を取得する
3. クラスターノードを適切にサイズ設定する
4. コンテナリクエストを適切にサイズ設定する
5. 使用率の低いノードを管理する
6. 中止されたワークロードを修復する
7. 推奨事項に基づいて行動する
8. セルフマネージドノードを更新する

環境に Kubecost をデプロイする

Amazon EKS Finhack Workshop では、 AWS 所有アカウントで Kubecost を使用するように設定された Amazon EKS 環境をデプロイする方法について説明します。これにより、テクノロジーを実際に体験することができます。組織でこのワークショップを実行することに関心がある場合は、アカウントチームにお問い合わせください。

Helm を使用して Kubecost を Amazon EKS クラスターにデプロイするには、 AWS AWS と Kubecost が共同で EKS のお客様にコストモニタリングを提供するブログ記事を参照してください。

あるいは、Kubecost のインストールと設定の手順については、Kubecost の公式ドキュメントを参照することもできます。Windows ノードの Kubecost サポートの詳細については、Kubecost ドキュメントの「Windows Node Support」を参照してください。

Windows アプリケーションの詳細なコスト内訳を取得する

Amazon EC2 スポットインスタンスを使用すると大幅なコスト削減を実現できますが、Windows ワークロードがステートフルである傾向があるという事実からもメリットを得ることができます。スポットインスタンスの使用はアプリケーションによって異なります。ユースケースに該当するかどうかを確認することをお勧めします。

Windows アプリケーションの詳細なコスト内訳を取得するには、Kubecost にログインします。ナビゲーションページで、[削減額] を選択します。

クラスターノードを適切にサイズ設定する

Kubecost で、ナビゲーションバーから [削減額] を選択し、[クラスターノードを適切なサイズにする] を選択します。

クラスターが vCPU と RAM の両方の点で過剰にプロビジョニングされていることを Kubecost が報告する例を考えてみましょう。次の表は、Kubecost の詳細と推奨事項を示しています。

	Current	推奨事項: シンプル	推奨事項: 複雑
合計数	1 か月あたり 3,462.57 USD	1 か月あたり 137.24 USD	1 か月あたり 303.68 USD
ノード数	4	5	4
CPU	74 個の vCPU	10 個の vCPU	8 個の vCPU
RAM	152 GB	20 GB	18 GB
インスタンスの内訳	2 個の c5.xlarge + 他 2 個	5 個の t3a.medium	2 個の c5n.large + 他 1 個

Kubecost ブログ記事「Find an optimal set of nodes for a Kubernetes cluster」で説明されているように、シンプルなオプションでは単一のノードグループを使用しますが、複雑なオプションではマルチノードグループのアプローチを使用します。[導入する方法] ボタンを使用すると、ワンクリックでク

ラスターのサイズ変更を実行できます。これには、Kubecost Cluster Controller のインストールが必要です。

eksctl によって作成されていないセルフマネージド型の Windows ノードを使用している場合は、「Updating an existing self-managed node group」を参照してください。これらの手順では、Auto Scaling グループで使用される Amazon EC2 起動テンプレートでインスタンスタイプを変更する方法を示します。

コンテナリクエストを適切にサイズ設定する

Kubecost で、ナビゲーションバーから [削減額] を選択し、[適切なサイズの推奨事項をリクエストする] ページに移動します。このページには、ポッドの効率、適切なサイズの推奨事項、および推定コスト削減額が表示されます。[カスタマイズ] ボタンを使用して、クラスター、ノード、名前空間\コントローラーなどでフィルタリングできます。

例えば、CPU と RAM (メモリ) の点で一部のポッドが過剰にプロビジョニングされていると Kubecost が計算したとします。その場合、Kubecost は、毎月の推定削減額を達成するために、新しい CPU 値と RAM 値に調整することを推奨します。CPU と RAM の値を変更するには、デプロイマニフェストファイルを更新する必要があります。

使用率の低いノードを管理する

Kubecost で、ナビゲーションバーから [削減額] を選択し、[使用率の低いノードを管理する] を選択します。

クラスター内の 1 つのノードが CPU と RAM (メモリ) の点で十分に使用されていないため、ドレインして終了するかサイズ変更できることがページに示されている例を考えてみましょう。ノードとポッドのチェックに合格しないノードを選択すると、ドレインできない理由の詳細が表示されます。

中止されたワークロードを修復する

Kubecost で、ナビゲーションバーから [削減額] を選択し、[中止されたワークロード] ページを選択します。この例では、windows と呼ばれる名前空間でフィルタリングします。このページには、トラフィックのしきい値を満たしておらず、中止されたと見なされるポッドが表示されます。ポッドは、定義された期間に一定量のネットワークトラフィックを送受信する必要があります。

1 つ以上のポッドが中止されていることを慎重に検討した後、レプリカの数をスケールダウンしたり、デプロイを削除したり、消費するリソースが少なくなるようにサイズを変更したり、デプロイが中止されたと思われることをアプリケーション所有者に通知したりすることで、コストを削減できます。

推奨事項に基づいて行動する

[クラスターノードを適切なサイズにする] セクションで、Kubecost は、クラスター内のワーカーノードの使用状況を分析し、コストを削減するためにノードの適切なサイズ設定に関する推奨を行います。Amazon EKS で使用できるノードグループには、セルフマネージド型とマネージド型の 2 つのタイプがあります。

セルフマネージドノードを更新する

セルフマネージド型ノードの更新については、Amazon EKS ドキュメントの「セルフマネージド型ノードの更新」を参照してください。eksctl を使用して作成されたノードグループは更新できず、新しい設定で新しいノードグループに移行する必要があることが示されています。

例えば、ng-windows-m5-2xlarge (m5.2xlarge EC2 インスタンスを使用する) という Windows ノードグループがあり、ポッドを ng-windows-t3-large (コストを削減するために t3.large EC2 インスタンスによってバックアップされる) という新しいノードグループに移行するとします。

eksctl によってデプロイされたノードグループを使用するときに新しいノードグループに移行するには、以下を実行します。

1. ポッドが現在存在するノードを見つけるには、kubectl describe pod <pod_name> -n <namespace> コマンドを実行します。
2. kubectl describe node <node_name> コマンドを実行します。出力は、ノードが m5.2xlarge インスタンスで実行されていることを示しています。また、ノードグループ名 (ng-windows-m5-2xlarge) とも一致します。
3. ノードグループ ng-windows-t3-large を使用するようにデプロイを変更するには、ノードグループ ng-windows-m5-2xlarge を削除して kubectl describe svc,deploy,pod -n windows を実行します。ノードグループが削除されたため、デプロイは直ちに再デプロイを開始します。

   > **Note**
   >
   > ノードグループを削除すると、サービスのダウンタイムが発生します。

4. 数分後に、kubectl describe svc,deploy,pod -n windows コマンドを再度実行します。出力は、ポッドがすべて再び実行中状態になっていることを示しています。
5. ポッドが現在ノードグループ ng-windows-t3-large で実行されていることを表示するには、kubectl describe pod <pod_name> -n <namespace> コマンドと kubectl describe node <node_name> コマンドを再度実行します。

代替のサイズ変更方法

この方法は、セルフマネージド型またはマネージド型ノードグループの任意の組み合わせに適用されます。ブログ記事「Seamlessly migrate workloads from EKS self-managed node group to EKS-managed node groups」には、ダウンタイムなしで、オーバーサイズのインスタンスタイプを持つ 1 つのノードグループから、適切にサイズ設定されたノードグループにワークロードを移行する方法に関するガイダンスが記載されています。

次の手順

Kubecost を使用すると、Amazon EKS 環境のコストを簡単に視覚化できます。Kubecost と Kubernetes および AWS APIs の深い統合は、潜在的なコスト削減を見つけるのに役立ちます。これらは、Kubecost の [削減額] ダッシュボードで推奨事項として確認できます。Kubecost は、クラスターコントローラー機能を通じて、これらの推奨事項の一部を実装することもできます。

「」のstep-by-stepデプロイAWS 」と「Kubecost collaborate to deliver cost monitoring for EKS customers」のブログ記事を AWS Containers ブログから参照することをお勧めします。

その他のリソース

- 「Amazon EKS Workshop」(Amazon EKS Workshop)
- AWS と Kubecost が共同で EKS 顧客のコストモニタリングを提供 (AWS ブログ)
- Amazon EKS Finhack Workshop (AWS Workshop Studio)
- での Windows コンテナ AWS (AWS Workshop Studio)

Windows アプリケーションを App2Container でリプラットフォームする

概要:

AWS App2Container は、Java および .NET ウェブアプリケーションをコンテナに移行してモダナイズするためのコマンドラインツールです。App2Container は、ベアメタル、仮想マシン、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、またはその他のクラウドプロバイダーで実行されているすべてのアプリケーションのインベントリを分析して構築します。コンテナ化するアプリケーションを選択します。App2Container は、アプリケーションアーティファクトと依存関係をコンテナイメージにパッケージ化し、ネットワークポートを設定し、必要な Amazon Elastic Container

Service (Amazon ECS) および Amazon Elastic Kubernetes Service (Amazon EKS) デプロイアーティファクト (Infrastructure as Code (IaC) テンプレート) を生成します。App2Container は、コンテナ化されたアプリケーションを本番環境にデプロイするために必要なクラウドインフラストラクチャと CI\CD パイプラインをプロビジョニングします。詳細については、App2Container ドキュメントの「How App2Container works」を参照してください。

App2Container を使用すると、アプリケーションのデプロイ AWS とオペレーションを標準化しながら、アプリケーションをコンテナとして移行してモダナイズできます。App2Container を使用して、概念実証 (PoC) を迅速に構築したり、コンテナへの本番ワークロードのデプロイを高速化したりできます。

Windows アプリケーションを使用する際は、いくつかの点に留意する必要があります。App2Container は、Microsoft Internet Information Services (IIS) にデプロイされた ASP.NET アプリケーションのコンテナ化をサポートしています。これには、Windows Server 2016、Windows Server 2019、または Windows Server Core 2004 で実行される、IIS がホストする Windows Communication Foundation (WCF) アプリケーションが含まれます。詳細については、App2Container ドキュメントの「Supported applications for Windows」を参照してください。App2Container は、コンテナアーティファクトのベースイメージとして Windows Server Core を使用し、Windows Server Core コンテナのバージョンを、コンテナ化コマンドを実行するサーバーのオペレーティングシステム (OS) のバージョンと一致させます。このアプローチでは、基盤となる OS からアプリケーションを切り離し、従来の移行を実行せずに OS をアップグレードできるようにします。

ワーカーマシンを使用してアプリケーションをコンテナ化する場合、Windows Server 2019 長期サービスチャネル (LTSC) などのコンテナベースイメージは、Windows Server 2019 などのワーカーマシンの OS と一致します。アプリケーションサーバーで直接コンテナ化を実行している場合、バージョンはアプリケーションサーバーの OS と一致します。アプリケーションが Windows Server 2008 または 2012 R2 で実行されている場合は、コンテナ化とデプロイのステップ用にワーカーマシンを設定することで、App2Container を引き続き使用できます。App2Container は、Windows 7 や Windows 10 などの Windows クライアントオペレーティングシステムで実行されているアプリケーションをサポートしていません。App2Container は、Java プロセス用の Tomcat、TomEE、および JBoss (スタンドアロンモード) フレームワークをサポートしています。詳細については、「App2Container の互換性」を参照してください。

コスト上の利点

アプリケーションをコンテナ化して統合すると、1 つのアプリケーションと 1 つのサーバーのデプロイ設計パターンと比較して、コンピューティングを最大 60% 節約できます。App2Container は、

アプリケーションのコンテナ化プロセスの迅速化に役立ちます。モダナイズのニーズに合わせて App2Container を使用する利点は次のとおりです。

- App2Container は、追加料金なしで提供されます。
- App2Container は、コンテナイメージ内の複数のアプリケーションをサポートします。
- App2Container を使用してレガシー .NET アプリケーションをコンテナに移行することで、サポート終了が近づいているオペレーティングシステムに対処します。新しいオペレーティングシステムに移行して、延長サポートの支払いを回避し、セキュリティリスクを軽減できます。
- コンテナは、.NET アプリケーションをパッケージ化する効率的で費用対効果の高い方法です。「MACO Recommendation - Moving to containers」でコンテナの利点を確認してください。
- アプリケーションの統合とコンテナ化により、コンピューティングリソースをより効率的に使用して、コンピューティング、ストレージ、およびライセンスのフットプリントを削減できます。
- コンテナに移行すると、運用上のオーバーヘッドとインフラストラクチャのコストを削減し、開発の移植性とデプロイの俊敏性を向上させることができます。

コスト最適化の推奨事項

App2Container の使用方法については、「AWS App2Containerの開始方法」を参照してください。App2Container コマンドの詳細については、「App2Containeコマンドリファレンス」を参照してください。

次の手順

App2Container は、アプリケーションをコンテナ化し、Amazon EKS または Amazon ECS にデプロイするプロセスを高速化できます。コンテナへのアプリケーションのデプロイにより、コンピューティング、ネットワーク、およびストレージのコストが削減され、アプリケーションオペレーターの運用オーバーヘッドが削減されます。

App2Container の実践的なエクスペリエンスについては、「Modernize with AWS App2Container Workshop」を参照してください。深層学習の経験が必要な場合は、 AWS アカウントチームに App2Container イマージョンデーを設定するよう依頼してください。

その他のリソース

- を使用した複雑な多層 Windows アプリケーションのコンテナ化 AWS App2Container (AWS ブログ記事)

- を使用したレガシー ASP.NET アプリケーションのコンテナ化 AWS App2Container (AWS ブログ記事)
- App2Container でサポートされているアプリケーション (AWS ドキュメント)
- AWS App2Container ワークショップによるモダナイズ (AWS Workshop Studio)
- AWS App2Container FAQs (AWS ウェブサイト)

ストレージ

Microsoft ワークロードに適したストレージを選択することは、アーキテクチャ上の重要な決定です。意思決定プロセスの一環として、ストレージプランを作成し、アプリケーションとサービスの機能要件を決定することをお勧めします。この章では、計画に含める可能性のある以下のストレージオプションの概要を説明します。

セクション:

- Amazon EBS
- Amazon FSx
- AWS Storage Gateway

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用できる永続的なブロックレベルストレージボリュームを保存できるフルマネージドブロックストレージサービスです。Amazon EBS のいくつかの機能を活用して、クラウド内の Windows ワークロードのストレージリソースを効果的に管理および最適化できます。例えば、Amazon EBS を使用してワークロードに必要な IOPS とスループットの正確な量をプロビジョニングし、ワークロードの要件に合わせてさまざまなボリュームタイプから選択し、ツールを使用して無駄なストレージリソースを特定して排除できます。このようにストレージのパフォーマンスと使用状況をきめ細かく制御することで、不要なコストを回避しながらストレージリソースを最適化できます。

このセクションでは、次のトピックについて説明します。

- Amazon EBS ボリュームを gp2 から gp3 に移行する
- Amazon EBS スナップショットを変更する
- アタッチされていない Amazon EBS ボリュームを削除する

Amazon EBS ボリュームを gp2 から gp3 に移行する

概要

ソリッドステートドライブ (SSD) は、本番および高パフォーマンスワークロード用の標準ストレージオプションです。Amazon EBS は、中 ~ 高パフォーマンスのワークロード用の汎用 SSD ボリュー

ムを提供します。多くの AWS のサービス (Amazon EC2 を含む) の標準は、これらの汎用 SSD ボリュームの第 2 世代である gp2 です。gp3 と呼ばれる第 3 世代の汎用 SSD は、2020 年 12 月にリリースされました。

gp3 オファリングは、前世代と比較してパフォーマンスのカスタマイズの側面を大幅に改善しました。Amazon EBS gp2 ボリュームの場合、パフォーマンスはボリュームのサイズと密接に関係しています。1 GB のキャパシティごとに、gp2 ボリュームは 3 IOPS のパフォーマンスを実現します。つまり、2,000 GB の gp2 ボリュームは 6,000 IOPS が実現可能です。gp3 ボリュームの場合、パフォーマンスはストレージキャパシティとは別にカスタマイズできます。これにより、少量の容量でも、最大 80,000 IOPS と 2,000 Mb/秒のスループットのパフォーマンス機能を実現できます。

gp3 ボリュームのもう 1 つの大きな変更点は、ベースライン IOPS パフォーマンスです。gp3 ボリュームは 3,000 IOPS から始まります。これに対して、gp2 ボリュームは、同じパフォーマンス能力に到達する前に 1 TiB のサイズに達する必要があります。通常 C: ドライブが 1 TiB よりはるかに小さい Windows Server の場合、gp2 から gp3 にアップグレードするとパフォーマンスが大幅に向上します。

最後に、gp3 ボリュームの価格は、gp2 ボリュームと比較して最も大きな改善点の 1 つです。gp3 ボリュームは、gp2 ボリュームよりも 20% 低コストで、強化されたパフォーマンス能力を提供します。

コストへの影響

パフォーマンスをキャパシティから独立してスケーリングできるので、IOPS とスループットを追加することの料金面を理解することが重要です。gp2 ボリュームの場合、料金は月あたり 0.10 USD/GiB のプロビジョンドキャパシティに基づきます。gp3 ボリュームの場合、料金は高パフォーマンスのプロビジョンド IOPS SSD ボリュームに似ています。キャパシティに対して 1 つのコストがかかり、追加の IOPS とスループットには別のコストがかかります。

次の表に示すように、gp3 ボリュームのキャパシティ料金は、月額 0.08 USD/GiB (gp2 より 20% 安価) です。個別のコストは、IOPS については、3,000 を超えるプロビジョンド IOPS ごとに月額 0.005 USD、スループットについては、125 MiB/秒を超えるプロビジョンド MiB/秒ごとに月額 0.04 USD です。

	gp3	gp2
ボリュームサイズ	1 GiB ~ 64 TiB	1 GiB – 16 TiB
ベースライン IOPS	3,000	3 IOPS/GiB (最小 100 IOPS) から最大 16,000 IOPS。

	gp3	gp2
		1 TiB 未満のボリュームは、最大 3,000 IOPS までバーストできます。
最大 IOPS/ボリューム	80,000	16,000
ベースラインスループット	125 MiB/秒	スループットの制限は、ボリュームサイズに応じて 128 MiB/秒〜250 MiB/秒です。
最大スループット/ボリューム	2,000 MiBs	250 MiB/秒
価格	0.08 USD/GiB/月 3,000 IOPS まで無料。3,000 を超えるプロビジョンド IOPS ごとに月額 0.005 USD。 125 MiB/秒まで無料。125 MiB/秒を超えるプロビジョンド MiB/秒ごとに月額 0.04 USD。	月あたり 0.10 USD/GiB

> **Important**
>
> gp3 ボリュームにはキャパシティとパフォーマンスに別々のコストがかかりますが、gp3 ボリュームが同じパフォーマンスレベルで設定されている場合は、gp2 ボリュームよりも常に安くなります。

次の表は、さまざまなキャパシティおよびパフォーマンス設定で gp2 ボリュームを gp3 ボリュームに変換することで実現できるコスト削減の例を示しています。

gp2 設定の例

ボリュームサイズ (GiB)	最大 IOPS	スループット (MiB/秒)	コスト (USD/月)
30	3000	128	3.00 USD
100	3000	128	10.00 USD
500	3000	250	50.00 USD
1,000	3000	250	100.00 USD
2000	6000	250	200.00 USD
6000	16000	250	600.00 USD

gp3 (ベースライン) 設定の例

最大 IOPS	スループット (MiB/秒)	コスト (USD/月)	コスト削減 (gp2 と比較)
3000	125	2.40 USD	20%
3000	125	8.00 USD	20%
3000	125	40.00 USD	20%
3000	125	80.00 USD	20%
3000	125	160.00 USD	20%
3000	125	480.00 USD	20%

gp3 (gp2 マッチング) 設定の例

最大 IOPS	スループット (MiB/秒)	コスト (USD/月)	コスト削減 (gp2 と比較)
3000	128	2.52 USD	16%

最大 IOPS	スループット (MiB/秒)	コスト (USD/月)	コスト削減 (gp2 と比較)
3000	128	8.12 USD	19%
3000	250	45.00 USD	10%
3000	250	85.00 USD	15%
6000	250	180.00 USD	10%
16000	250	550.00 USD	8%

コスト分析については、「Amazon EBS のリソース」の「EBS gp2 から gp3 への移行コスト削減計算ツール」セクションを参照してください。この計算ツールをダウンロードして使用すれば、gp2 ボリュームを gp3 ボリュームに移行することでどれだけ節約できるか調べることができます。

コスト最適化の推奨事項

移行プロセスを完了する方法については、 AWS ストレージブログの「Amazon EBS ボリュームを gp2 から gp3 に移行し、コストを最大 20% 削減する」を参照してください。

その他のリソース

- Amazon EBS ボリュームを gp2 から gp3 に移行し、コストを最大 20% 削減 (AWS ストレージブログ)
- Amazon EBS ボリュームタイプを最適化するための AWS Config カスタムルールの構築 (AWS クラウドオペレーションと移行ブログ)
- 未使用の Amazon EBS ボリュームを削除して AWS コストを制御する (AWS クラウドオペレーションと移行ブログ)
- 「Amazon EBS Migration Utility」(GitHub)
- 「Finding savings from 2020 re:Invent announcements」(AWS クラウド財務管理)
- コスト最適化ワークショップ (AWS Well-Architected Labs)
- gp2 から gp3 への移行で削減できるコストの計算ツール (ダウンロード)

Amazon EBS スナップショットを変更する

概要

EBS ボリュームを削除し、スナップショットの保持とアーカイブを管理することは、最初からコストを制御する上で重要な側面です。ポイントインタイムスナップショットを作成することで、EBS ボリューム上のデータを、Amazon Simple Storage Service (Amazon S3) にバックアップできます。スナップショットは増分バックアップです。つまり、最後のスナップショット以降に変更されたデバイス上のブロックのみが保存されます。これにより、スナップショットを作成するのに要する時間が最小限に抑えられ、データを複製しないことで、ストレージコストが節約されます。各スナップショットには、(スナップショットを作成したときから) データを新しい EBS ボリュームに復元するために必要な情報がすべて含まれます。

EBS スナップショットの料金は、ギガバイト/月単位で計算されます。スナップショットのサイズと保持期間に対して請求されます。料金はストレージ層によって異なります。標準階層の場合は、保存されている変更されたブロックに対してのみ請求されます。アーカイブ階層の場合は、保存されているすべてのスナップショットブロックに対して請求されます。また、アーカイブ階層からスナップショットを取得する場合も請求が発生します。各ストレージ階層のシナリオの例を次に示します。

- 標準階層 – 100 GB のデータを保存するボリュームがあります。最初のスナップショット (スナップ A) の 100 GB の全データに対して請求されます。次のスナップショット (スナップ B) の時点では、105 GB のデータがあります。この場合、増分スナップ B の追加ストレージ 5 GB に対してのみ請求されます。
- アーカイブ階層 – スナップ B をアーカイブします。スナップショットはアーカイブ階層に移動され、105 GB のスナップショットブロック全体に対して課金されます。

Amazon Data Lifecycle Manager を使用すると、スナップショットをスケジュールどおりに保持および管理するためのライフサイクルを設定できます。

コストへの影響

EBS ボリュームとスナップショットの料金は個別に管理されます。EBS スナップショットは、アクティブな EBS ボリュームよりも低いレートで請求されます。インスタンスが終了すると、アタッチされた各 EBS ボリュームの DeleteOnTermination 属性の値を使用して、ボリュームを保持するか削除するかを決定します。デフォルトでは、ルートボリュームの `DeleteOnTermination` 属性は `True` に設定されます。他のすべてのボリュームタイプでは `False` に設定されます。これにより、オペレーターが EC2 インスタンスを削除しようとしているが、ルートボリュームに加えてインスタ

ンスに追加されたボリュームが残ってしまう状況が発生します。不要になったボリューム (および関連するスナップショット) を確認する手順については、Amazon EBS ドキュメントの「Amazon EBS ボリュームに関する情報の表示」を参照してください。

デフォルトでは、スナップショットを作成すると、Amazon EBS スナップショットスタンダード階層に保存されます (標準階層)。標準階層に保存されるスナップショットは、増分です。これは、最新のスナップショットが作成された後に変更された、ボリューム内のブロックのみが保存されることを意味します。Amazon EBS Snapshots Archive は、頻繁な検索や高速な検索を必要としない、アクセス頻度の低いスナップショットを低コストで長期保存するために使用できる新しいストレージ階層です。標準とアーカイブの料金の違いは大きく、スナップショット戦略を設定する際に重要な考慮事項となるはずです。Amazon EBS Snapshots Archive では、90 日以上保存する予定で、ほとんどアクセスする必要のないスナップショットに対して、スナップショットストレージコストを最大 75% 削減できます。

Amazon EBS スナップショットストレージ	Cost
標準	月額 0.05 USD/GB
アーカイブ	月額 0.0125 USD/GB

小規模な環境では、コスト削減はそれほど大きくない場合があります。EBS ボリュームが削除された場合でも、複数のアカウントと数千の EC2 インスタンスがあり、数 TB の EBS スナップショットが存在するような大規模な環境では、コスト削減はさらに大きくなります。

次の表は、使用量がわずか 50 TB の、1 か月あたりの標準階層とアーカイブ階層を比較したものです。この低い規模でも、年間数千ドルの節約になります。

Amazon EBS スナップショットストレージ	1 か月あたりのコスト	1 年あたりのコスト
標準 50 TB	312.50 USD	3,750 USD
アーカイブ 50 TB	78.13 USD	937.60 USD
	年間削減額	2,812.40 USD

コスト最適化の推奨事項

スナップショットを削除しても、組織のデータストレージコストが減少しない場合があります。他のスナップショットはそのスナップショットのデータを参照する場合があります。参照されたデータは常に保持されます。たとえば、最初にデータボリュームが 10 GiB のスナップショットを取得した場合は、スナップショットのサイズも 10 GiB になります。スナップショットは増分バックアップであるため、2 回目に同じボリュームのスナップショットを取得した場合は、最初のスナップショットを取得後に変更したデータブロックのみ含まれます。2 回目のスナップショットで、最初のスナップショットのデータを参照することもできます。4 GiB のデータを変更し、2 回目のスナップショットを取得した場合、2 番目のスナップショットのサイズは 4 GiB になります。また、2 番目のスナップショットでは、最初に取得した、変更していない 6 GiB のスナップショットを参照できます。詳細については、「EBS ボリュームのスナップショットを削除してからボリューム自体を削除した後、ストレージコストが削減されなかったのはなぜですか？」を参照してください。 AWS ナレッジセンターの「」を参照してください。

以下の点を考慮してください。

- 別の が AWS アカウント 所有し、アカウントと共有しているスナップショットに対しては課金されません。共有スナップショットをお客様のアカウントにコピーした場合にのみ請求されます。また、共有スナップショットから作成した EBS ボリュームに対しても請求されます。
- スナップショット (スナップ A) が別のスナップショット (スナップ B) によって参照されている場合、スナップ B を削除してもストレージコストは削減されない可能性があります。スナップショットを削除すると、そのスナップショットに固有のデータだけが削除されます。他のスナップショットによって参照されるデータは残り、この参照されたデータに対して請求されます。増分スナップショットを削除するには、Amazon EBS ドキュメントの「Incremental snapshot deletion」を参照してください。

スナップショットのクリーンさは、 AWSでワークロードを実行する際の標準的な運用方法です。時間の経過とともに、スナップショットが、不要なデータに対する高額なコストを発生させる可能性があります。

その他のリソース

- 未使用の Amazon EBS ボリュームを削除して AWS コストを制御する (AWS クラウドオペレーションと移行ブログ)
- 「Delete an Amazon EBS snapshot」(Amazon EBS ドキュメント)
- コスト最適化ワークショップ (AWS Well-Architected Labs)

- 「Automatically archive Amazon EBS Snapshots with Amazon Data Lifecycle Manager」(AWS ストレージブログ)

アタッチされていない Amazon EBS ボリュームを削除する

概要

アタッチされていない (孤立した) EBS ボリュームは、 AWS 環境に不要なストレージコストが発生する可能性があります。 AWS 環境衛生の一環として、未使用および未使用の EBS ボリュームの定期的なレビューと削除を組み込むことが重要です。EBS ボリュームの使用状況を継続的に確認するプロセスを整えることがベストプラクティスです。AWS Compute Optimizer を使用して、十分に活用されていないインスタンスを確認できます。このセクションでは、アタッチされていない、または十分に活用されていない EBS ボリュームを特定、管理、削除するのに役立ちます。

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにストレージボリュームを提供するブロックレベルのデバイスです。EBS は、EC2 インスタンスをアタッチおよびデタッチできる柔軟性を備えた永続的ストレージを提供します。つまり、EC2 インスタンスが終了した場合でも、EBS ボリュームのライフサイクルは維持されます。DeleteOnTermination 属性は、インスタンスの終了時に、アタッチされた EBS ボリュームを保持するか削除するかを制御する機能です。デフォルトでは、ルートボリュームの属性は `True` に設定され、削除が実行されます。他のボリュームでは `False` に設定され、保持が実行されます。

コストへの影響

アタッチされていない EBS ボリュームは、未使用ボリュームまたは孤立ボリュームとも呼ばれ、プロビジョニングされたストレージサイズとストレージタイプに基づいて、アタッチされたボリュームと同じ料金が発生します。Amazon EBS 料金の平均コストは月額 0.10 USD/GB と最小限に見えるかもしれませんが、未使用の EBS ボリュームが蓄積されると、時間の経過とともに大きなコストにつながる可能性があることを認識することが重要です。

例えば、次の表に示すように、それぞれが 100 GB のストレージサイズでプロビジョニングされた未使用の EBS ボリュームを 50 個保持した場合の影響を考えてみましょう。

ストレージボリュームの数	ボリュームタイプ	サイズ	合計月額コスト
50 ボリューム	gp2 (0.10 USD)	100 GB	100 GB 50.00 EBS ボリューム 0.10 USD/月 = 500.00 USD

上記の表のシナリオでは、1 か月あたり約 500 USD、つまり年間 6,000 USD のコスト削減となります。これはコスト削減に向けた効果的なステップです。アタッチされていない EBS ボリュームの削除は、 AWS 環境衛生の通常のプラクティスとして組み込んでください。

コスト最適化の推奨事項

を使用して AWS 、アタッチされていない EBS ボリュームの削除を簡単に自動化できます。例えば AWS Lambda、、 AWS Config、Amazon CloudWatch、 を使用して、年齢、タグ、その他の仕様に基づいて、アタッチされていないボリュームを削除する基準 AWS Systems Manager を定義できます。これらを使用して AWS のサービス 、クリーンアッププロセスを大規模に自動化することもできます。

意図しない結果を避けるため、アタッチされていない EBS ボリュームを削除する前にデューデリジェンスを実行することをお勧めします。

アタッチされていない EBS ボリュームを管理する

次のベストプラクティスを検討することをお勧めします。

- コンプライアンス要件を満たす – アタッチされていない EBS ボリュームの削除が組織のガバナンスおよびコンプライアンス要件に準拠していることを確認します。
- データバックアップおよび保持ポリシーを設定する – アタッチされていない EBS ボリュームを削除する前に、重要なデータを別のストレージリポジトリ (Amazon S3 など) にバックアップします。データ保持に関しては、Amazon EBS スナップショットは EBS ボリュームよりもコスト効率の高いデータ保持方法であり、将来必要に応じてボリュームを復元できます。スナップショットの効果的な管理の詳細については、このガイドの「Amazon EBS スナップショットを変更する」セクションを参照してください。
- 依存関係をチェックする – アタッチされていない EBS ボリュームと他の AWS リソース間の依存関係をチェックします。AWS マネジメントコンソール または API を使用して、サイズ、ステータス、関連リソースなど、EBS ボリュームに関する説明情報を収集できます。これは、一時的にアタッチされていないリソースを削除しないようにするための重要なステップです。

- 保持ポリシーを作成する – アタッチされていない EBS ボリュームの保持期間を設定します。これにより、アタッチされていないボリュームを削除する適切な時間を特定し、 AWS 環境を最適化し続けることができます。例えば、スケジュールに基づいて Lambda 関数を開始する Amazon EventBridge ルールを作成できます。Lambda 関数は AWS SDK を使用して、アタッチされていない EBS ボリュームをアクティブに識別し、タグ付けメカニズムを適用して追跡を容易にし、アタッチされていない EBS ボリュームが定義されたしきい値に達したか超えたときに通知を送信できます。
- アタッチされていない EBS ボリュームにタグを付ける – EBS ボリュームのタグ付けは、環境、アプリケーション、所有者などの属性に基づいてボリュームを整理および識別するのに役立つ便利な方法です。これは、アタッチされていないボリュームのどれを削除するかを決定するときに特に役立ちます。タグに基づいて、不要になったボリュームをすばやく特定できるためです。
- 安全な削除を確保する – EBS ボリュームが最後にアタッチされた日時を確認すれば、ボリュームを削除しても安全かどうかを判断するのに役立ちます。詳細については、AWS CLI 「コマンドを使用して特定の Amazon EBS ボリュームの添付ファイルまたはデタッチ履歴を一覧表示するにはどうすればよいですか？」を参照してください。 AWS ナレッジセンターの 。
- 使用率の低い EBS ボリュームの特定 – 使用率の低い EBS ボリュームの特定と削除は、ストレージコストを削減し、最適化された AWS 環境を維持するために強く推奨されるプラクティスです。AWS Trusted Advisor と AWS Compute Optimizerは、使用率の低い EBS ボリュームを特定し、コストを削減して効率を向上させるための推奨事項を提供するのに役立ちます。例えば、 (GitHub) を使用した EBS ボリュームの最適化の自動化の設定 AWS Trusted Advisor、Trusted Advisor 組織 (TAO) ダッシュボードの確立 (AWS Workshop Studio)、および (AWS ストレージブログ) を使用した Amazon EBS ボリュームのコスト最適化 AWS Compute Optimizerを参照してください。

アタッチされていない EBS ボリュームのクリーニングを自動化する

アタッチされていない EBS ボリュームのクリーニングを自動化するには、次のツールを検討することをお勧めします。

- AWS APIs (DescribeVolumes) – AWS SDKs または AWS Command Line Interface () を使用して、アタッチされていない EBS ボリュームをフィルタリングして見つけることができますAWS CLI。スケジュールに従って実行されるスクリプトまたは Lambda 関数を使用してこのプロセスを自動化することで、時間と労力を節約できます。GitHub のサンプルスクリプトは、これがどのように動作するかを示しています。このスクリプトは Lambda を使用して AWS CloudTrail ログを分析し、アタッチされていない EBS ボリュームを識別します。
- AWS Systems Manager 自動化 – これにより、インフラストラクチャの定期的なメンテナンスと修復タスクを自動化できます。開始するには、自動化ランブックを作成します。このランブック

で、特定の順序で実行される一連のステップを定義します。例えば、最初にアタッチされていない EBS ボリュームのスナップショットを作成し、次にボリューム自体を削除するランブックを作成できます。これにより、手動で実行すると時間がかかり、エラーが発生しやすいタスクを自動化できます。

- AWS Config – これにより、 AWS リソースへの経時的な変更を評価、監査、追跡できます。設定変更をキャプチャすることで、 AWS Config を使用して環境のコンプライアンス、ガバナンス、リソース使用率を評価できます。たとえば、 は未使用の EBS ボリュームを識別 AWS Config できます。さらに、 AWS Systems Manager オートメーションを に関連付ける AWS Config ことで、未使用の EBS ボリュームの削除を自動的に修正できます。

その他のリソース

- AWS Config と を使用して未使用の Amazon Elastic Block Store (Amazon EBS) ボリュームを削除する AWS Systems Manager (AWS 規範ガイダンス)
- 未使用の Amazon EBS ボリュームを削除して AWS コストを制御する (AWS クラウドオペレーションと移行ブログ)
- 「AWSConfigRemediation-DeleteUnusedEBSVolume」(AWS Systems Manager Automation ランブックリファレンス)

Amazon FSx

Amazon FSx for Windows File Server は、Windows ワークロード用に最適化されたフルマネージドファイルストレージサービスです。複雑なストレージインフラストラクチャ管理を必要とせずに、Windows ベースのアプリケーションとワークロードを実行するためのシンプルでスケーラブルなソリューションを提供します。FSx for Windows File Server を使用すると、Microsoft SQL Server、Microsoft SharePoint、カスタム .NET アプリケーションなど、Windows アプリケーションをネイティブにサポートする共有ファイルストレージを簡単にプロビジョニングしてアクセスすることができます。さらに、FSx for Windows File Server は、従量制料金やストレージクォータなどの柔軟な料金オプションと、ストレージフットプリントを削減し、パフォーマンスとコストを最適化するための自動データ重複排除を提供することで、コストの管理を支援します。

このセクションでは、次のトピックについて説明します。

- 適切な SMB ファイルストレージを選択する
- Amazon FSx でデータ重複排除を有効にする
- FSx for Windows File Server のデータシャーディングを理解する

- Amazon FSx での HDD ボリュームの使用状況を理解する
- 単一のアベイラビリティーゾーンを使用する

適切な SMB ファイルストレージを選択する

概要

AWS は、最新の AWS インフラストラクチャイノベーションとセキュリティを組み合わせながら、業界をリードするファイルサービスの豊富な機能を提供するさまざまなフルマネージドストレージサービスを提供します。 AWS サービスを Infrastructure as Code (IaC) ワークフローに組み込み、 AWS コンピューティング、モニタリング、およびデータ保護サービスと統合することができます。Windows ワークロードでは、アプリケーションのニーズに合わせて使用できる 2 つのフルマネージドファイルサービス、すなわち FSx for Windows File Server と Amazon FSx for NetApp ONTAP から選択できます。

FSx for Windows File Server

Amazon FSx for Windows File Server は、Windows Server 上に構築されたフルマネージド共有ストレージを提供し、幅広いデータアクセス、データ管理、および管理機能を提供します。FSx for Windows File Server は Windows ネイティブサービスであるため、Windows 環境と簡単に統合できます。ユーザーおよびグループ共有には FSx for Windows File Server を使用し、SQL Server、Windows アプリケーション、仮想デスクトップインフラストラクチャ (VDI) には Always On フェイルオーバークラスターインスタンスを使用することをお勧めします。FSx for Windows File Server は、Amazon FSx File Gateway、Amazon Kendra、Amazon S3 の監査ログ、Amazon Data Firehose とも適切に統合されます。

FSx for ONTAP

FSx for ONTAP は、NetApp 独自の ONTAP ファイルシステムに基づいています。ある程度のスキルアップが必要であり、主に既存のオンプレミス NetApp ユーザーに対して推奨されます。一般的なユースケースには、ユーザーおよびグループ共有、SQL Server の Always On フェイルオーバークラスターインスタンス、Windows アプリケーションなどがあります。FSx for ONTAP は、複数のプロトコル、64 TB を超えるファイルシステム (DFS 名前空間サーバーを使用しない PB スケール)、クローン作成、レプリケーション、スナップショット、圧縮 (ストレージ効率)、データのインテリジェントな階層化をサポートしています。

コストへの影響

FSx for Windows File Server

FSx for Windows File Server は、SQL Server のフェイルオーバークラスターインスタンスをデプロイ AWS するための最初の共有ストレージソリューションでした。FSx for Windows File Server では、SQL Standard Edition ライセンスを使用してフェイルオーバークラスターインスタンスを起動できます。ただし、これにより、SQL Server Enterprise Edition ライセンスを必要とする Always On 可用性グループに依存することができなくなります。SQL Server Enterprise Standard Edition から SQL Server Standard Edition に切り替えることで、SQL Server ライセンスを 65～75% 節約できます。

フェイルオーバークラスターインスタンスに FSx for Windows File Server を使用して、一般的な EBS ストレージからストレージ I/O をオフロードできます。I/O を FSx for Windows File Server にオフロードすることで、ストレージのスループットに影響を与えることなく、高い Amazon EBS スループットと IOPS に依存する EC2 インスタンスをスケールダウンできます。

FSx for ONTAP

FSx for ONTAP を使用してブロックプロトコル iSCSI で Microsoft フェイルオーバークラスターを実行すると、SQL Server のインスタントファイル初期化、SnapMirror を使用したクロスリージョンレプリケーション、ウイルス対策サポート、クローン作成のメリットが得られます。テスト用のデータベースのコピーを複数作成する場合、クローンを作成すると、スペースの消費とそれらのデータベースコピーの作成速度の両方に大きな違いが生じる可能性があります。さらに、NetApp SnapCenter を使用すると、FSx for ONTAP を使用して SQL Server の EC2 インスタンスでバックアップ、復元、およびクローン機能を管理できます。FSx for ONTAP は、パフォーマンスとコスト効率のために、SSD から低コストのキャパシティプールストレージへの自動階層化も提供します。

FSx for ONTAP は、Windows ネイティブ NTFS ファイルシステムをサポートする FSx for Windows File Server とは異なり、NetApp のファイルシステム (ONTAP) をサポートしています。FSx for ONTAP の最小サイズは 1024 GB ですが、FSx for Windows File Server は 32 GB から開始できます。

Microsoft 分散ファイルシステムとの統合

FSx for Windows File Server と FSx for ONTAP は Microsoft の分散ファイルシステム (DFS) と統合され、既存のデプロイにシームレスに統合されます。アーキテクチャを計画するときは、次の点に注意してください。

- FSx for Windows File Server と FSx for ONTAP は、両方のデプロイタイプ (複数のアベイラビリティーゾーンと単一のアベイラビリティーゾーン) で DFS 名前空間 (DFSN) をサポートしています。
- DFS レプリケーション (DFSR) をサポートするのは FSx for Windows File Server のみであり、単一のアベイラビリティーゾーンを使用する場合のみです。

コスト最適化の推奨事項

FSx for Windows File Server と FSx for ONTAP の両方のパフォーマンスは、料金と同様に、設定に大きく依存します。FSx for Windows File Server の料金は、主にストレージキャパシティとストレージタイプ、スループットキャパシティ、バックアップ、および転送されたデータによって異なります。FSx for ONTAP では、SSD ストレージ、SSD IOPS、キャパシティプールの使用状況、スループットキャパシティ、バックアップに対して料金が発生します。

ファイルサービス	5 TB ストレージのコスト	設定	リージョン
FSx for Windows File Server	982.78 USD	単一のアベイラビリティーゾーン SSD (15,000 IOPS) 32 MBps 5 TB バックアップ (重複排除による節約なし)	米国東部 (バージニア北部)
FSx for ONTAP	979.28 USD	単一のアベイラビリティーゾーン 100% SSD 15,000 読み取り/書き込みキャパシティ階層 15,000 SSD IOPS	米国東部 (バージニア北部)

ファイルサービス	5 TB ストレージのコスト	設定	リージョン
		128 MBps 5 TB バックアップ (重複排除による節約なし)	

以下に留意してください。

- 重複排除と圧縮により、データサイズを縮小することで物理デバイスにより多くのデータを保存できますが、プロビジョニングされたソリッドステートドライブ (SSD) またはハードディスクドライブ (HDD) ストレージに対して料金が発生します。
- FSx for ONTAP を使用してデータを階層化できます。100% のデータが定期的にアクセスされ、SSD ストレージが必要になることは非常にまれです。コールドデータやアクセス頻度の低いデータをキャパシティ階層に移動してコストを削減できます。
- ここで説明する料金は、SSD 階層の 100% データおよび SSD 階層の 15,000 IOPS で計算されます。

バックアップ

デフォルトでは、FSx for ONTAP と FSx for Windows File Server の両方がフルマネージドバックアップを Amazon S3 に保存します。ただし、FSx for ONTAP では、SnapVault を使用してバックアップするための追加オプションがあります。これにより、キャパシティ階層に存在するようにバックアップを設定できます。SnapVault でのバックアップは、デフォルトのフルマネージドバックアップオプションよりもコスト効率の高いセルフマネージドメカニズムです。フルマネージドバックアップオプションは月額 0.05 USD/GB です。FSx for ONTAP の SnapVault バックアップ (SSD とキャパシティプールストレージの比率 10:1) は 0.03221 USD (0.9x0.0219+0.1x0.125) です。

以下に留意してください。

- AWS マネージドバックアップは、1 時間の粒度を提供します。SnapVault を使用すると、5 分単位にまで短縮できます。
- NetApp のツール (CLI や API など) を使用して、SnapVault の関係とスナップショットレプリケーションを設定できます。

- SnapVault ボリュームの all 階層化ポリシーを有効にして、キャパシティ階層をバックアップデータのストレージとして使用します。
- SnapVault の送信先は、同じ AWS リージョン、クロスリージョン、またはオンプレミスにすることができます。これは通常、単一のアベイラビリティーゾーンまたは複数のアベイラビリティーゾーンのファイルシステムのバックアップ先です。これに対して、 AWS Backup は Amazon S3 のリージョンの耐障害性に基づいています。

適切なサイジング

また、適切なサイジングと、過剰なプロビジョニングの防止を行うことで、コストを削減し、ファイルシステムを最大限に活用することもできます。

適切なサイズにするには、以下を実行します。

1. データに基づいて現在のニーズを特定します。一般的な Windows ワークロードでは、パフォーマンスモニターなどの組み込みオペレーティングシステムツールを使用できます。
2. パフォーマンスモニターで、次のカウンターを使用して現在のパフォーマンスニーズを測定します。キャプチャ間隔は 1 秒に設定され、最大ログサイズは 1,000 MB で、上書きが有効になります。

```
Logman.exe create counter PerfLog-Short -o "c:\perflogs\PerfLog-Long.blg" -f bincirc
 -v mmddhhmm -max 1024 -c "\LogicalDisk(*)\*" "\Memory\*" "\.NET CLR Memory(*)\*"
 "\Cache\*" "\Network Interface(*)\*" "\Paging File(*)\*" "\PhysicalDisk(*)\*"
 "\Processor(*)\*" "\Processor Information(*)\*" "\Process(*)\*" "\Thread(*)\*"
 "\Redirector\*" "\Server\*" "\System\*" "\Server Work Queues(*)\*" "\Terminal
 Services\*" -si 00:00:01
```

3. ログキャプチャを開始するには、logman start PerfLog-Short コマンドを実行します。ログキャプチャを停止するには、logman stop PerfLog-Short コマンドを実行します。

> **Note**
>
> パフォーマンスログファイルは、キャプチャを実行しているサーバーの c:\perflogs にあります。詳細については、Microsoft ドキュメントの「Windows Performance Monitor Overview」を参照してください。

4. 正しい設定を特定したら、Microsoft DISKSPD などのディスクストレスツールを使用して、Amazon FSx ファイルシステムで見積もりが正しいかどうかをテストします。
5. パフォーマンスに満足したら、ファイル共有にカットオーバーします。

ストレージキャパシティはスケールアップしかできないため、控えめなアプローチをお勧めします。スループットキャパシティは、必要に応じてスケールアップまたはスケールダウンできます。

その他のリソース

- Amazon FSx for NetApp ONTAP FAQs (AWS ウェブサイト)
- 新しいメトリクスによる Amazon FSx for Windows File Server のパフォーマンスの最適化 (AWS ストレージブログ)

Amazon FSx でデータ重複排除を有効にする

概要

データ重複排除は、データをより効率的に、より少ないキャパシティ要件で保存できる機能です。これには、データの忠実度や完全性を損なうことなく、データ内の重複を検出して削除することが含まれます。データ重複排除は、サブファイル可変サイズのチャンキングと圧縮を使用します。これにより、一般的なファイルサーバーでは 2:1、仮想化データでは最大 20:1 の最適化比が実現します。データ重複排除は、NTFS 圧縮よりもはるかに効果的です。重複排除アーキテクチャに内在するのは、ハードウェア障害時の耐障害性です。メタデータや最もアクセス頻度の高いデータチャンクの冗長性など、データとメタデータの完全なチェックサム検証が行われます。

FSx for Windows File Server は、データ重複排除を完全にサポートしています。これを使用すると、汎用ファイル共有の平均削減率が 50～60% になります。共有内では、ユーザードキュメントの場合は 30～50%、ソフトウェア開発データセットの場合は最大 70～80% が節約範囲です。データ重複排除によって達成できるストレージの節約は、ファイル間で重複する量など、データセットの性質によって異なることを理解する必要があります。保存されるデータが本質的に動的である場合、重複排除は適切なオプションではありません。

コストへの影響

企業におけるデータストレージの増加に対応するために、管理者はサーバーを統合し、キャパシティスケーリングとデータ最適化を主要な目標とします。データ重複排除のデフォルト設定により、すぐに削減を実現できます。また、管理者が設定をファインチューニングしてさらなるメリットを得ることもできます。例えば、特定のファイルタイプでのみ実行するように重複除外を設定したり、カスタムジョブスケジュールを作成したりできます。

大まかに言うと、重複排除には最適化、ガベージコレクション、スクラブの 3 種類のジョブがあります。最適化後にガベージコレクションジョブを実行するまで、スペースは解放されないことに注意

してください。ジョブをスケジュールすることも、手動で実行することもできます。データ重複排除ジョブをスケジュールするときに使用できるすべての設定は、ジョブを手動で開始するときにも使用できます (スケジュール固有の設定を除く)。

重複排除による実質的な削減は 25% に過ぎないとしても、FSx for Windows File Server では大幅なコスト削減が実現します。これらの予測される削減額は、 AWS 料金見積りツールの見積もりに基づいています。

コスト最適化の推奨事項

FSx for Windows File Server ファイルシステムの重複排除は、デフォルトでは有効になっていません。PowerShell のリモート管理を使用して重複排除を有効にするには、`Enable-FSxDedup` コマンドを実行してから `Set-FSxDedupConfiguration` コマンドを使用して設定を行う必要があります。詳細については、FSx for Windows File Server ドキュメントの「Administering file systems」を参照してください。

重複排除を有効にするには、次のコマンドを実行します。

```
PS C:\Users\Admin> Invoke-Command -ComputerName amznfsxzzzzzzzz.corp.example.com -
ConfigurationName FSxRemoteAdmin -ScriptBlock {Enable-FsxDedup }
```

重複排除の設定を確認するには、次のコマンドを実行します。

```
Invoke-Command -ComputerName amznfsxzzzzzzzz.corp.example.com -ConfigurationName
 FSxRemoteAdmin -ScriptBlock {
Set-FSxDedupSchedule -Name "CustomOptimization" -Type Optimization -Days
 Mon,Tues,Wed,Sat -Start 09:00 -DurationHours 7
}
```

PowerShell `Measure-DedupFileMetadata` コマンドレットを実行することで、フォルダのグループ、1 つのフォルダ、または 1 つのファイルを削除し、ガベージコレクションジョブを実行した場合に、ボリューム上でどれだけのディスクスペースを再利用できるかを特定できます。具体的に言うと、`DedupDistinctSize` 値が、これらのファイルを削除した場合にどれだけのスペースが戻るかを示します。多くの場合、ファイルには他のフォルダ間で共有されるチャンクがあるため、重複排除エンジンは、ガベージコレクションジョブ後に削除される一意のチャンクを計算します。

デフォルトのデータ重複排除ジョブスケジュールは、推奨ワークロードに対して適切に動作し、可能な限り非侵入的であるように設計されています (バックアップ使用タイプで有効になっている優先最適化ジョブを除く)。ワークロードのリソース要件が大きい場合は、アイドル時間中にのみジョブの

実行をスケジュールするか、データ重複排除ジョブが消費できるシステムリソースの量を減らすか増やすことをお勧めします。

デフォルトでは、データ重複排除は使用可能なメモリの 25% を使用します。ただし、これは `-memory switch` を使用して増やすことができます。最適化ジョブの場合は、15～50 の範囲を設定することをお勧めします。スケジュールされたジョブでは、より高いメモリ消費量を使用できます。例えば、ガベージコレクションジョブとスクラブジョブ (通常、オフ時間で実行するようにスケジュール) では、より高いメモリ消費量 (50 など) を設定できます。

データ重複排除設定の詳細については、FSx for Windows File Server ドキュメントの「データ重複排除によるストレージコストの削減」を参照してください。

その他のリソース

- 「Understanding Data Deduplication」(Microsoft ドキュメント)
- 「Reducing storage costs with Data Deduplication」(FSx for Windows File Server ドキュメント)

FSx for Windows File Server のデータシャーディングを理解する

概要

FSx for Windows File Server のパフォーマンスは設定によって異なります。主にストレージタイプ、ストレージキャパシティ、およびスループット設定に基づいています。選択したスループットキャパシティによって、ネットワーク I/O 制限、CPU とメモリ、ファイルサーバーによって課されるディスク I/O 制限など、ファイルサーバーで使用できるパフォーマンスリソースが決まります。選択したストレージキャパシティとストレージタイプによって、ストレージボリュームで使用できるパフォーマンスリソース、つまりストレージディスクによって課されるディスク I/O 制限が決まります。パフォーマンスに加えて、設定の選択もコストに影響します。FSx for Windows File Server の料金は、主にストレージキャパシティとストレージタイプ、スループットキャパシティ、バックアップ、および転送されたデータによって異なります。

ファイルストレージとパフォーマンスの要件が比較的大きい場合は、データシャーディングのメリットを受けることができます。データシャーディングには、より小さなデータセット (シャード) にファイルデータを分割し、それらを異なるファイルシステム間に保存することが含まれています。複数のインスタンスからデータにアクセスするアプリケーションは、これらのシャードに対して並行して読み取りと書き込みを行うことで、高いレベルのパフォーマンスを設定できます。同時に、共通の名前空間下で統一されたビューをアプリケーションに表示することもできます。また、大きいファ

イルデータセット (最大数百ペタバイト) に対して各ファイルシステムがサポートするサイズ (64 TB) を超えるファイルのデータストレージをスケーリングする際にも役立ちます。

コストへの影響

大規模なデータセットの場合、通常、同じレベルのパフォーマンスを実現するために、1 つの大きな SSD 共有ではなく、複数の小さな FSx for Windows File Server ファイルシステムをデプロイする方が効果的です。FSx for Windows File Server HDD ストレージタイプと SSD ストレージタイプを組み合わせて使用すると、コストをさらに削減でき、ワークロードを基盤となる最適なディスクサブシステムに適合させることができます。次の表では、単一の 17 TB ファイルシステムの違いを確認し、同じキャパシティに追加する複数の小さなファイルシステムと比較することができます。

複数のワークロードを持つ大規模な SSD ファイルシステム

[Server name] (サーバー名)	Cost	設定	リージョン
Amazon FSx for Windows File Server	5,716 USD	17 TB SSD 30% 重複排除 256 Mbps 17 TB バックアップ	米国東部 (バージニア北部)

DFSN を使用したパーティション化されたワークロード

[Server name] (サーバー名)	Cost	設定	リージョン	共有
Amazon FSx for Windows File Server	1,024 USD	2 TB SSD 20% 重複排除 128 Mbps 2 TB バックアップ	米国東部 (バージニア北部)	共有 1

[Server name] (サーバー名)	Cost	設定	リージョン	共有
		マルチ AZ		
Amazon FSx for Windows File Server	2,132 USD	5 TB SSD 30% 重複排除 256 Mbps 5 TB バックアップ マルチ AZ	米国東部 (バージニア北部)	共有 2
Amazon FSx for Windows File Server	1,036 USD	10 TB HDD 40% 重複排除 128 Mbps 10 TB バックアップ マルチ AZ	米国東部 (バージニア北部)	共有 3
DFSN Windows EC2 インスタンス	27 USD	t3a.medium 2 vCPU 4 GiB メモリ	米国東部 (バージニア北部)	DFSN インスタンス

大規模な SSD ファイルシステムの年間コストは 68,592 USD です。パーティション化されたワークロードの年間コストは 50,640 USD です。この例では、ワークロードを適切なバックエンドストレージに適合させながら、26% の削減を実現できます。料金見積もりの詳細については、AWS 料金見積りツール の見積もりを参照してください。

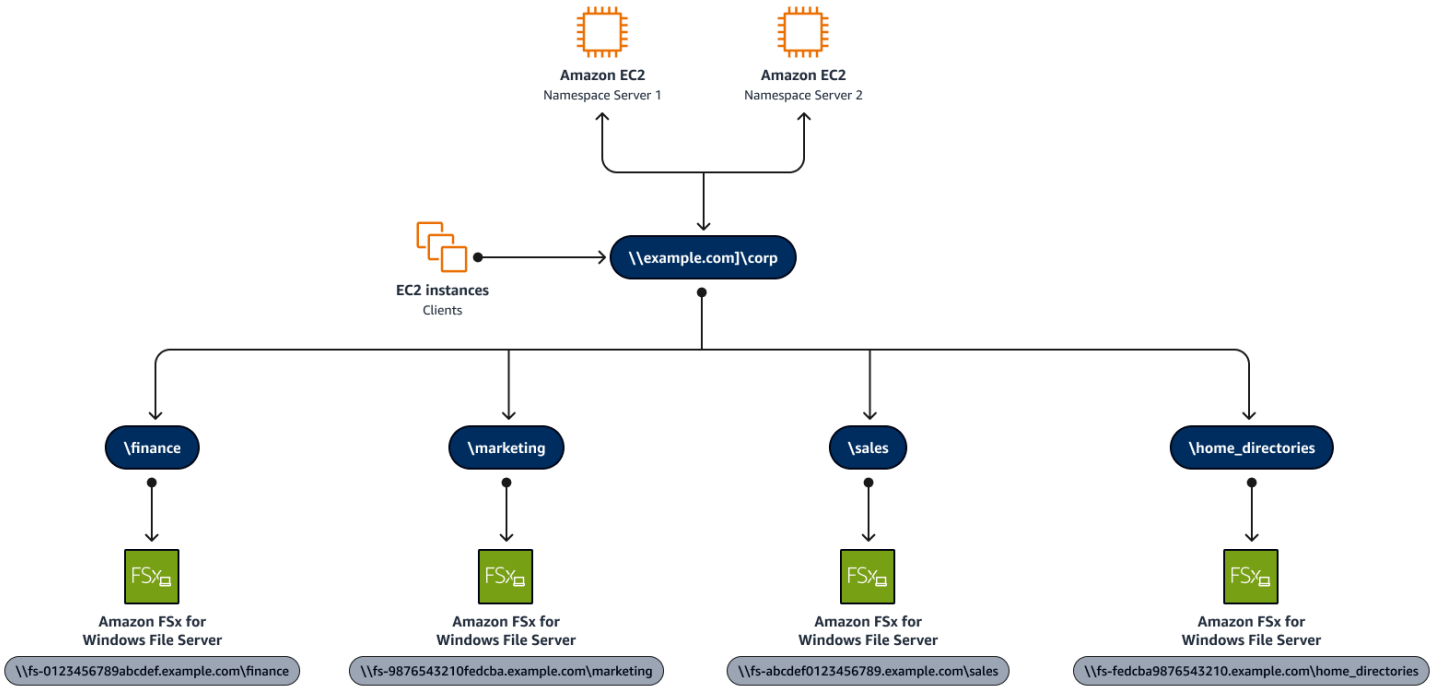
コスト最適化の推奨事項

データ重複排除ソリューションをデプロイするには、データのタイプ、I/O サイズ、I/O アクセスパターンに基づいて Microsoft DFS 名前空間を設定する必要があります。各名前空間は、最大 50,000 のファイル共有と、全体で数百ペタバイトのストレージキャパシティをサポートします。

使用する予定のすべてのファイルシステムに I/O を均等に分散するシャーディング方式を選択すると、最も効率的に機能します。ワークロードのモニタリングは、追加の最適化やコスト削減に役立ちます。Amazon FSx ファイルシステムのパフォーマンス情報の測定にヘルプが必要な場合は、FSx for Windows File Server ドキュメントの「FSx for Windows File Server performance」を参照してください。

シャーディング戦略を選択したら、DFS 名前空間を使用して共有に簡単にアクセスできるようにファイルシステムをグループ化できます。これにより、ユーザーは、実際には専用のユースケースを使用してさまざまなファイルシステムにアクセスしているときも、1 つの同種ファイルシステムを見ることになります。エンドユーザーが共有がどのようなワークロード向けに設計されているかを簡単に解読できるように、適切な命名規則で共有を作成することが重要です。また、エンドユーザーが誤って間違ったファイルシステムにファイルを配置しないように、本番用共有と非本番用共有にラベルを付けることも重要です。

次の図は、単一の DFS 名前空間を複数の Amazon FSx ファイルシステムのアクセスポイントとして使用する方法を示しています。



以下に留意してください。

- 既存の FSx for Windows File Server 共有を DFS ツリーに追加できます。
- Amazon FSx を DFS 共有パスのルートに追加することはできません。サブフォルダは 1 つだけです。
- DFS 名前空間設定を提供するには、EC2 インスタンスをデプロイする必要があります。

DFS-N 設定の詳細については、Microsoft ドキュメントの「DFS Namespaces overview」を参照してください。DFS 名前空間の使用の詳細については、YouTube の動画「Using DFS Namespaces with Amazon FSx for Windows File Server」を参照してください。

その他のリソース

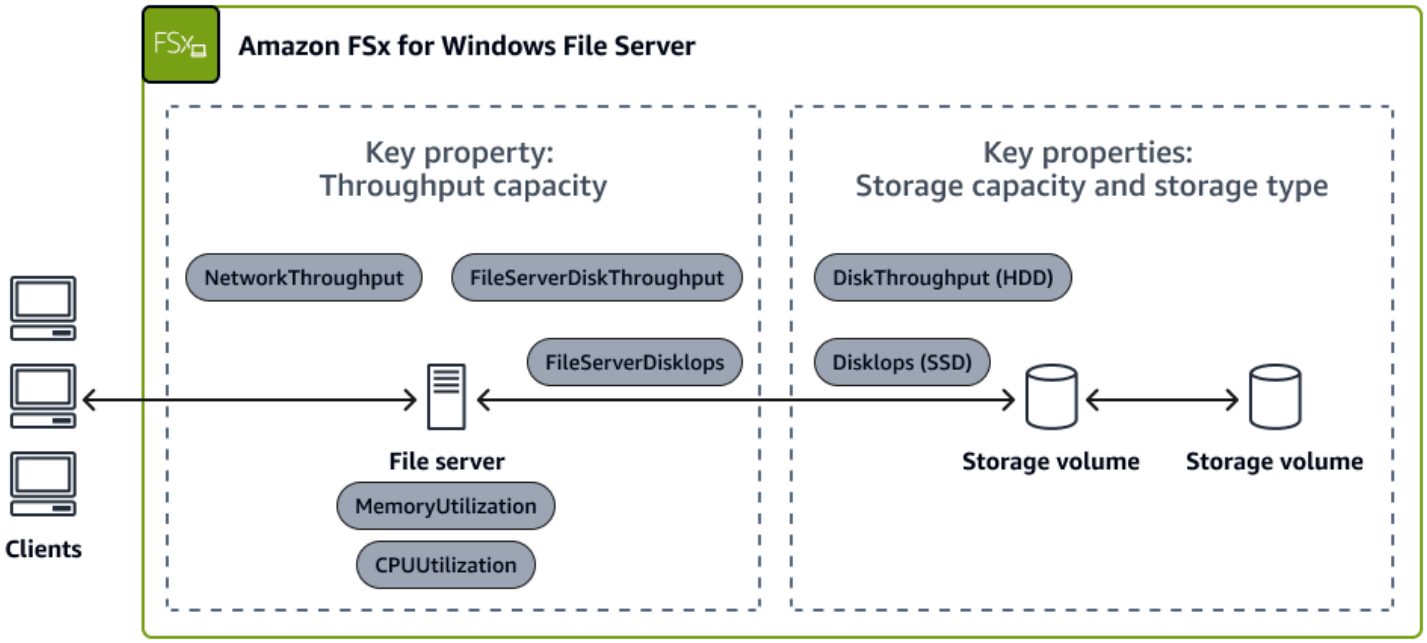
- 「Grouping multiple file systems with DFS Namespaces」(Amazon FSx ドキュメント)
- 「Walkthrough 6: Scaling out performance with shards」(Amazon FSx ドキュメント)
- 「Using DFS Namespaces with Amazon FSx for Windows File Server」(AWS ラボ)

Amazon FSx での HDD ボリュームの使用状況を理解する

概要

Amazon FSx for Windows File Server では、ファイルシステムのキャパシティに関係なくスループットを柔軟に選択できます。ハードディスクドライブ (HDD) とソリッドステートドライブ (SSD) の 2 つのキャパシティ設定を使用できます。

次の図は、スループットとストレージ設定の関係を示しています。



HDD ベースのストレージでは、80 バーストディスク IOPS (ストレージ 1 TiB あたりの IOP) の 12 IOPS ベースラインと、80 バーストメガバイト/秒 (ストレージ 1 TiB あたり) の 12 メガバイト/秒 ベースラインのスループットが得られます。例えば、共有のサイズが 50 TB の場合、スループットと IOPS の両方のベースラインとして 50 * 12 = 600 が得られます。

Amazon FSx for Windows File Server は 80 バースト IOPS を提供します。バーストクレジットは、使用率がベースラインレートを下回ると自動的に補充され、使用率がベースラインレートを上回ると自動的に消費されます。例えば、ワークロードが 1 時間に 10 IOPS/TB しか使用していない場合 (ベースラインレートより 2 IOPS/TB 低い)、バーストクレジットが再び不足するまで、次の 1 時間は 14 IOPS/TB (ベースラインより 2 IOPS/TB 高い) を使用できます。

ファイル操作の場合、Amazon FSx for Windows File Server は SSD ストレージで一貫したミリ秒未満のレイテンシーを提供し、HDD ストレージで 1 桁のミリ秒のレイテンシーを提供します。HDD ストレージを備えたファイルシステムを含むすべてのファイルシステムにおいて、Amazon FSx for Windows File Server はファイルサーバーに高速 (インメモリ) キャッシュを提供するため、ストレージタイプに関係なく、アクティブにアクセスされたデータに対して高いパフォーマンスとミリ秒未満のレイテンシーを実現できます。

必要に応じて、HDD ストレージを使用することで、全体的なストレージキャパシティのコストを削減し、ニーズに合わせて信頼性の高いストレージプラットフォームを提供できます。

コストへの影響

Amazon FSx for Windows File Server のパフォーマンスは、ストレージキャパシティ、ストレージタイプ、スループットの 3 つの要因によって決まります。ネットワーク I/O パフォーマンスとインメモリキャッシュサイズはスループットキャパシティによってのみ決まり、ディスク I/O パフォーマンスはスループットキャパシティ、ストレージタイプ、ストレージキャパシティの組み合わせによって決まります。

I/O 負荷の高いワークロードには SSD が推奨されますが、HDD パフォーマンス仕様でニーズを満たすことができるさまざまなワークロードがあります。HDD ストレージは、ホームディレクトリ、ユーザーおよび部門の共有、コンテンツ管理システムなど、幅広いワークロード向けに設計されています。例えば、ユーザーが現在のプロジェクトをサポートするデータへの低レイテンシーアクセスのみを必要とする場合、保存しているデータのほとんどは頻繁にアクセスされません。

AWS 料金見積りツール を使用して、20 TB SSD と `us-east-1` の HDD ファイルシステムを比較できます。次の表に示すように、重複排除による節約がない場合でも、HDD ファイルシステムと SSD ファイルシステムを比較すると、コストの差は大きいです。

Amazon FSx ファイルシステム設定	月額コスト
20 TB マルチ AZ SSD (`us-east-1`)	4,699.30 USD
20 TB マルチ AZ HDD (`us-east-1`)	542.88 USD
月間削減額の見積もり	4,156.42 USD

> **Note**
>
> FSx for Windows File Server のその他の節約については、このガイドの「Amazon FSx でデータ重複排除を有効にする」セクションを参照してください。

パフォーマンスニーズを正しく特定することで、ワークロードに適したストレージを選択し、コストを削減することができます。

コスト最適化の推奨事項

HDD ストレージを使用する場合は、ファイルシステムをテストして、パフォーマンス要件を満たすことを確認してください。HDD ストレージは SSD ストレージに比べて低コストですが、ストレー

ジ単位あたりのディスクスループットとディスク IOPS のレベルが低くなります。I/O 要件の低い汎用のユーザー共有やホームディレクトリ、データの取得頻度が低い大規模なコンテンツ管理システム、またはサイズの大きいファイルの数が少ないデータセットに適する場合があります。

既存のファイルシステムのストレージタイプは変更できません。Amazon FSx for Windows File Server ファイルシステムのストレージタイプを変換するには、既存のファイルシステムをバックアップし、目的のストレージタイプの新しいファイルシステムに復元する必要があります。既存の SSD ファイルシステムを HDD ファイルシステムに変換する場合は、HDD の最小キャパシティが 2 TB とかなり大きいことに注意してください。

異なるストレージタイプでバックアップを復元するには、以下を実行します。

1. 既存のファイルシステムをバックアップします。
2. HDD ストレージタイプで新しい Amazon FSx ファイルシステムを作成します。
3. 希望するストレージタイプを使用して、バックアップを新しいファイルシステムに復元します。
4. 新しいファイルシステムのストレージタイプが正しく、データがそのままであることを確認します。

変更を本番環境に移行する前に、Amazon FSx ファイルシステムのパフォーマンスを分析し、変更が許容できるかどうかを確認することをお勧めします。詳細については、「 AWS ストレージブログ」の「新しいメトリクスを使用した Amazon FSx for Windows File Server のパフォーマンスの最適化」の投稿を参照してください。

その他のリソース

- 「Optimizing costs with Amazon FSx」(Amazon FSx ドキュメント)

単一のアベイラビリティーゾーンを使用する

概要

このセクションでは、Amazon FSx for Windows File Server の単一のアベイラビリティーゾーン実装を使用する方が有益な場合について説明します。ここでは、単一のアベイラビリティーゾーンに移行することでコストを削減しながら、マネージドファイルストレージサービスとして Amazon FSx for Windows File Server を使用できるようにするシナリオについて説明します。本番用ワークロードには、Amazon FSx の単一のアベイラビリティーゾーンを実装することをお勧めします。これにより、複数のアベイラビリティーゾーンの冗長性を確保できます。

コストへの影響

単一のアベイラビリティーゾーンファイルシステムは、複数のアベイラビリティーゾーンの実装と比較して約 40% のコスト削減を実現します。複数のアベイラビリティーゾーンファイルシステムの場合、SSD では月額 0.230 USD/GB、HDD では月額 0.025 USD/GB を支払うのに対し、単一のアベイラビリティーゾーンファイルシステムの場合は、SSD では月額 0.130 USD/GB、HDD では月額 0.013 USD/GB を支払うことになります。AWS 料金見積りツール を使用して、コストの比較を表示し、独自の見積もりを作成することができます。

10 TB ファイルシステムの場合は、複数のアベイラビリティーゾーンに対して月額約 1,200 USD を支払うか、単一のアベイラビリティーゾーンに対して月額 680 USD を支払うかの違いになります。この例では、SSD で 10 TB の FSx for Windows File Server ファイルシステムを使用しています。重複排除による推定削減は 50% です。全体として、単一のアベイラビリティーゾーンのエントリコストは低くなりますが、次のセクションで説明するいくつかの注意点があります。

コスト最適化の推奨事項

単一のアベイラビリティーゾーンのデプロイ

単一のアベイラビリティーゾーンが適切であることを確認するには、FSx for Windows File Server に保存されるデータに対する独自の内部 SLA を考慮してください。これには、顧客 (内部および外部) に提供する SLA があるかどうか、Amazon FSx 単一アベイラビリティーゾーンの 99.9% の可用性でそれらの SLA を満たすことができるかどうかの理解が必要です。単一のアベイラビリティーゾーンを持つ FSx for Windows File Server のアップタイムは依然として 99.9% です。複数のアベイラビリティーゾーンに対する Amazon FSx の SLA は 99.99% を超えています。ミッションクリティカルなワークロードの場合は、追加コストがかかっても、単一のアベイラビリティーゾーンではなく複数のアベイラビリティーゾーンを使用することをお勧めします。

単一のアベイラビリティーゾーンのデプロイは、SQL Server データベースのバックアップなどのワークロードに最適です。HDD 階層で低コストのストレージを提供しながら、継続的なアップタイムを実現できます。可用性の高い SQL サーバーや本番用アプリケーションアクセスなど、本番用ワークロードに対してより高いレベルの可用性が必要な場合、単一のアベイラビリティーゾーンはワークロードに適していません。バックアップ、非本番のテスト、開発環境では、Amazon FSx 単一アベイラビリティーゾーン実装により運用コストを削減できます。

Amazon FSx 単一アベイラビリティーゾーンファイルシステムがうまく機能するユースケースの 1 つは、Always On 可用性グループを使用する高可用性 SQL Server クラスター内のサーバーごとのストレージとして、複数の Amazon FSx 単一アベイラビリティーゾーンファイルシステムが使用され

ている本番環境です。詳細については、 AWS ストレージブログの記事の「高可用性 SQL Server デプロイのコストの最適化 AWS」を参照してください。

マルチリージョンレプリケーション

単一アベイラビリティーゾーンファイルシステム (単一アベイラビリティーゾーンファイルシステムのみが機能するもの) でコストを削減するための潜在的なオプションは、Amazon FSx でマルチリージョンレプリケーションを利用する場合です。ネイティブ Microsoft DFS-R での使用をサポートするシングル AZ ファイルシステムをデプロイできます。DFS-R には、リージョンや複数のサイトにまたがってデータを自動的にレプリケートする機能があります。Amazon FSx を使用して DFS-R を設定する方法の詳細については、Amazon FSx ドキュメントの「Using Microsoft Distributed File System Replication」を参照してください。

マルチリージョンのコスト削減のもう 1 つの方法は、 を使用することです AWS Storage Gateway。これにより、Amazon FSx のマルチリージョンアクセスのために Amazon FSx ファイルゲートウェイを別のリージョンに実装できるようになります。詳細については、このガイドの「AWS Storage Gateway」セクションを参照してください。

リージョン間で作業する場合は、クロスリージョンのデータトラフィックのデータ転送コストを考慮する必要があります。リージョン間を移動するトラフィックには、0.02 USD/Gb の料金が発生します。したがって、大量のデータ変更が継続的に発生する場合は、全体的なコストが増加します。例として、1 TB のデータ転送は約 20.48 USD に相当します。

メンテナンスウィンドウ

Amazon FSx で単一のアベイラビリティーゾーンを使用している場合、メンテナンスウィンドウは重要な考慮事項です。メンテナンスウィンドウ中、基盤となる Windows Server の定期的なソフトウェアパッチ適用により、Amazon FSx ファイルシステムは約 20 分間使用できなくなります。夜間バックアップにファイルシステムを使用している場合は、バックアップ中の中断を避けるために適宜 Amazon FSx メンテナンスウィンドウを調整します。Amazon FSx ファイルシステムを作成した後、メンテナンスウィンドウを調整できます。

その他のリソース

- 「Availability and durability: Single-AZ and Multi-AZ file systems」(Amazon FSx ドキュメント)
- Amazon FSx for Windows File Server の料金 (AWS ウェブサイト)

AWS Storage Gateway

AWS Storage Gateway は、オンプレミス環境をクラウドストレージに接続するハイブリッド AWS クラウドストレージサービスです。これにより、既存のオンプレミスインフラストラクチャを とシームレスに統合できるため AWS、クラウドからデータを保存および取得し、ハイブリッド環境でアプリケーションを実行できます。Windows ワークロードの場合、Storage Gateway を使用し、SMB や NFS などのネイティブ Windows プロトコルを使用することでデータにアクセスしたりデータを保存したりすることができます。Storage Gateway を使用すると、オンプレミスのハードウェアとソフトウェアをクラウドへのブリッジとして使用 AWS することで、 での Windows ワークロードの実行に関連するコストを削減できます。これにより、既存のインフラストラクチャを大幅に変更 AWS することなく、 のスケーラビリティとコスト効率を活用できます。

Storage Gateway には、Amazon S3 File Gateway、Amazon FSx File Gateway、テープゲートウェイ、およびボリュームゲートウェイが含まれています。S3 File Gateway と FSx File Gateway は、Microsoft ワークロードで最も一般的に使用されます。

Amazon S3 ファイルゲートウェイ

Amazon S3 File Gateway を使用すると、従来の SMB 共有を使用してユーザーにアクセス権を提供しながら、ファイルを Amazon S3 に保存できます。これにより、使い慣れたユーザーインターフェイスが提供され、Amazon S3 にデータを保存し、さまざまな Amazon S3 ストレージ階層を活用することでコストを削減できます。S3 Intelligent Tiering で Storage Gateway を実装すると、ライフサイクルファイルを最小コストのストレージ階層に自動的に移動して、コストをさらに削減することができます。スケールアウト、読み取り専用アクセス、高速反復読み取り (キャッシュから)、およびデータベースダンプには、S3 File Gateway をお勧めします。一般的に、高パフォーマンスまたは高可用性の書き込み、ファイルの編集、部門の共有にはお勧めしません。

Amazon FSx ファイルゲートウェイ

Amazon FSx File Gateway も、Amazon FSx Windows ファイルシステムを使用する際のコスト削減を実現します。FSx File Gateway を立ち上げて、別のリージョンの Amazon FSx ファイルシステムへのローカライズされたアクセスを提供して、独立した 2 つのファイルシステムを持つコストを回避できます。これは、複数のオンプレミスファイルサーバーがあり、それらを統合して複数のハードウェアデバイスへの支払いを回避する場合にも役立ちます。

コストへの影響

Amazon S3 ファイルゲートウェイ

Storage Gateway の起動ウィザードを使用できるため、S3 File Gateway の設定は簡単です。 AWS 環境で EC2 インスタンスを使用することで、数分でゲートウェイをデプロイできます。ゲートウェイを設定したら、SMB および NFS プロトコルを介してアクセスできるように Storage Gateway 共有を設定できます。一般的な Windows ワークロードでは、この設定を使用して Active Directory 環境を活用し、ファイル共有に対するアクセス許可を設定することもできます。Storage Gateway は一般的な Windows ファイル共有として機能するため、通常の使用状況に効果的に統合できます。ファイルとフォルダはオブジェクトとして保存され、NTFS アクセスコントロールリスト (ACL) はメタデータとして保存されます。

次の表は、10 TB のストレージのコストと 3 つの使用可能なストレージオプションを比較したものです。

- FSx for Windows File Server
- Amazon S3 ファイルゲートウェイ
- Amazon Elastic Block Store (Amazon EBS)

Amazon S3 を使用すると、データをさまざまな使用階層に分割できるため、10 TB のストレージの保存料金が大幅に安くなります。料金見積もりでは、S3 Intelligent Tiering が料金の柔軟性のために使用されます。これには、S3 Standard の 80%、Infrequent Access の 10%、Amazon Glacier の 10% が含まれます。Amazon Glacier を使用できますが、Amazon Glacier に移動したファイルにすぐにアクセスする必要がないように、適切なライフサイクルルールを設定することが重要です。Amazon Glacier は、通常のアクセス用途ではなく、アーカイブ用途専用です。

ストレージシステム	10 TB のストレージのコスト	リージョン
FSx for Windows File Server (重複排除で 50% の節約を想定)	683.20 USD SSD	米国東部 (バージニア北部)
Amazon S3 ファイルゲートウェイ	449.51 USD Intelligent Tiering	米国東部 (バージニア北部)
Amazon EBS	1,335.69 USD GP3	米国東部 (バージニア北部)

以下の点を考慮してください。

- Amazon Glacier では、RestoreObject API を使用してオブジェクトを Amazon S3 に復元しない限り、汎用 I/O エラーが発生します。Amazon CloudWatch Events を使用して、この I/O エラーの通知を使用することをお勧めします。これにより、運用チームは、アクセスする必要があるファイルでこのエラーが発生したユーザーに応答できます。これらのエラーの詳細については、Amazon S3 File Gateway ドキュメントの「Error: InaccessibleStorageClass」を参照してください。
- Amazon Glacier のアクセス制限に加えて、Storage Gateway でオブジェクト/フォルダごとに許可される ACL は 10 個のみです。Storage Gateway を使用する前に、10 を超える ACL エントリが必要でないことを確認してください。

Amazon FSx ファイルゲートウェイ

Amazon S3 File Gateway と同様に、FSx File Gateway は、データを長期間保持するファイルシステムへのアクセスを提供します。Amazon S3 File Gateway では、データは Amazon S3 にあります。FSx File Gateway の場合、データは FSx for Windows File Server にあります。マルチ AZ オプションは FSx for Windows File Server で使用できますが、マルチリージョンオプションはありません。グローバル企業またはリモートオフィスがある場合は、レイテンシーを回避するために、地理的にエンドユーザーに近い共有ストレージプラットフォームを提供する必要がある場合があります。別の Amazon FSx ファイルシステムをデプロイする場合、まったく新しい Amazon FSx for Windows File Server ファイルシステムと必要なストレージのコストが追加されます。まったく新しいファイルシステムの作成やコストの重複を避けるために、FSx File Gateway をセカンダリリージョンにデプロイできます。これにより、ユーザーはファイルへのローカライズされたアクセスを利用できるようになると同時に、全体的なコストを削減できます。

ストレージシステム	10 TB のストレージのコスト	リージョン
Amazon FSx for Windows File Server	683.20 USD SSD	米国東部 (バージニア北部)
Amazon FSx ファイルゲートウェイ	503.70 USD/単一ゲートウェイ	米国東部 (バージニア北部)

> **Note**
>
> 上記の表の料金は、Storage Gateway の料金に基づいています。

以下に留意してください。

- FSx File Gateway を使用すると、マルチリージョンワークロードで 1 か月あたり約 180 USD (または年間 2,100 USD) を節約できます。
- FSx File Gateway では、定期的にアクセスされるファイルをキャッシュするだけで済み、完全なセカンダリコピーをキャッシュする必要がないため、データ転送料金ははるかに低くなります。
- 異なるリージョンに FSx for Windows File Server を 2 回デプロイし、 AWS Backup または で更新できますが AWS DataSync、どちらのオプションもほぼリアルタイムではありません。

コスト最適化の推奨事項

Amazon S3 ファイルゲートウェイ

S3 File Gateway は、ファイルを保存するための低コストのオプションを提供しますが、ファイルシステムの実装方法と使用方法に関して考慮すべき問題がいくつかあります。例えば、S3 File Gateway では、Storage Gateway ソフトウェアを実行するために仮想マシンを使用する必要があります。では AWS、Storage Gateway はデフォルトで m5.xlarge インスタンスを使用して Amazon EC2 にデプロイされます。オンプレミスのストレージコストを削減する場合は、VMware や Hyper-V などの仮想化プラットフォーム上に仮想アプライアンスとして Storage Gateway をデプロイできます。

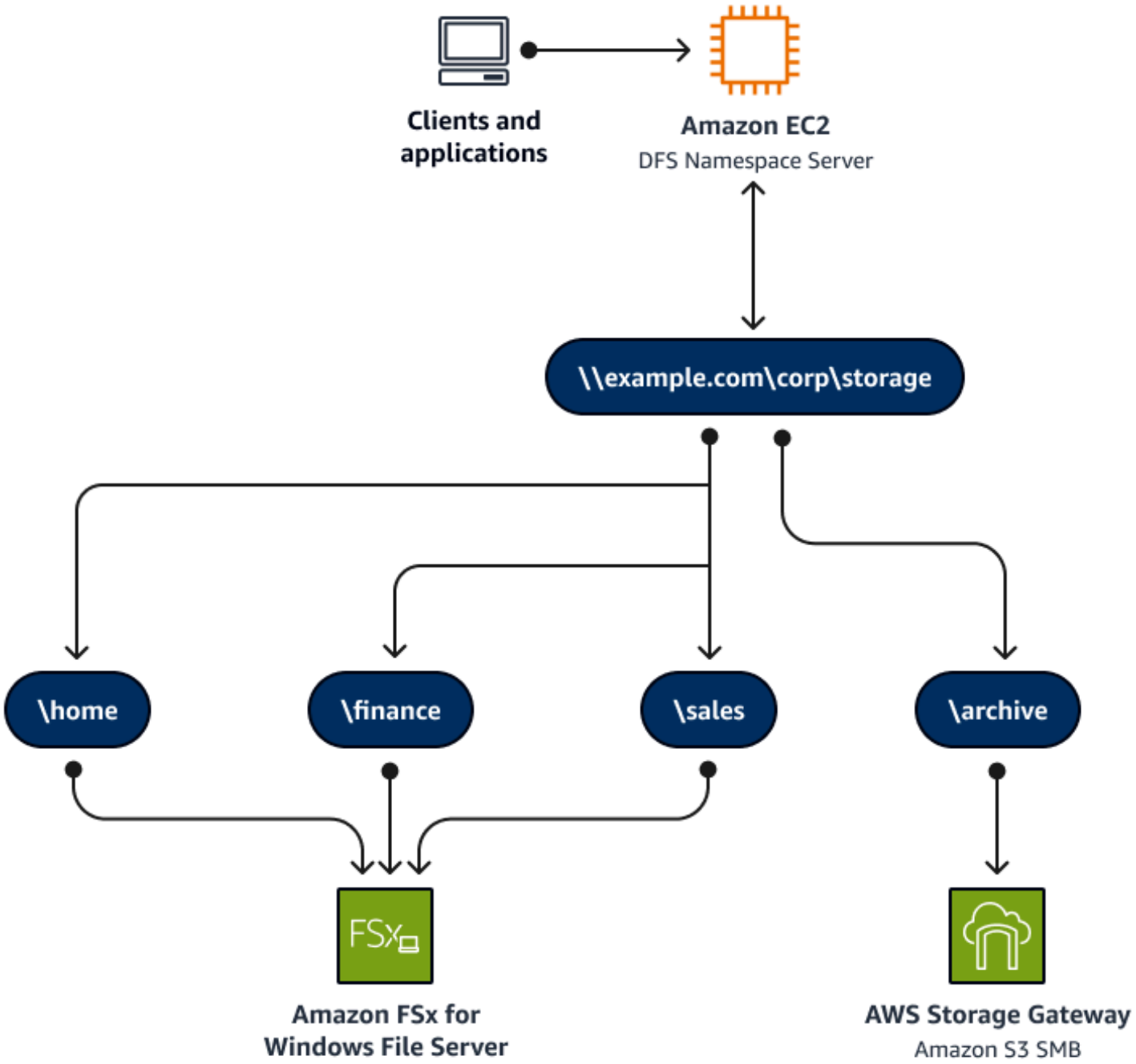
高可用性に関する考慮事項

Storage Gateway の実行は、ファイルへのアクセスの単一障害点です。不要なダウンタイムを防ぐため、Storage Gateway インスタンスを変更したり、停止/起動したりできるユーザーに対して厳格なアクセス制御を実装することをお勧めします。さらに、 でのデプロイでは AWS、Amazon Data Lifecycle Manager を使用してルーティングスナップショットを作成し、Storage Gateway の実装をすばやく復旧することをお勧めします。VMware を使用してオンプレミスで Storage Gateway を実行している場合は、高可用性を実現できるように設定できます。

複数のファイルシステムの実行

日常使用ファイルワークロードをアーカイブワークロードから分離すると、不要なストレージコストを回避できます。Storage Gateway は、FSx for Windows File Server ファイルシステムと一緒にデプロイできます。DFS 名前空間を使用すると、FSx for Windows File Server で実行されているプライマリ日常使用ストレージと、Amazon S3 で実行されているストレージ (Storage Gateway を介してアクセス) を提示できます。

次の図は、1 つの DFS 名前空間を、さまざまなバックエンドストレージオプションのフロントエンドアクセスポイントとして使用する方法を示しています。



クライアントは、\\example.com\storage などのフォルダ構造に移動します。このメインディレクトリには、サブディレクトリが含まれています。FSx for Windows File Server ファイルシステムには、通常アクセスされるファイル共有が含まれています。Storage Gateway で作成されたファイル共有をアーカイブデータに使用できます。ユーザーは項目を手動でアーカイブフォルダにアーカイブできま

す。また、通常のファイル共有からアーカイブフォルダへの一部のファイルの移動を自動化するプロセスを構築することもできます。

以下の点を考慮してください。

- ストレージ要件を確認し、キャッシュに適切なストレージを提供します。
- ゲートウェイを Active Directory 設定に追加し、標準の Windows ACL を使用してファイルにアクセスします。

FSx ファイルゲートウェイ

FSx File Gateway のデプロイは S3 File Gateway のデプロイに似ていますが、起動ウィザードを使用するとさらに簡単になります。詳細な手順については、Amazon FSx File Gateway ドキュメントの「Step 3: Create and activate an Amazon FSx File Gateway」を参照してください。環境に FSx File Gateway をデプロイしたら、既存の Amazon FSx ファイルシステムに関連付けて、ファイルにアクセスすることができます。

ストレージは、FSx File Gateway をデプロイする際の主な考慮事項です。デフォルトのストレージは 150 GB で、これはファイルをキャッシュするのに十分なスペースです。フリースペースが少ない場合のモニタリングアラートを作成すると、ストレージのサイズを過剰割り当てなしで適切に設定するのに役立ちます。

その他のリソース

- AWS Storage Gateway リソース (AWS ドキュメント)

Active Directory

Windows Server を実行する Amazon Elastic Compute Cloud (Amazon EC2) は、Windows ベースのアプリケーションとワークロードをデプロイするための安全で信頼性が高く、高性能な環境です。インスタンスを迅速にプロビジョニングし、必要に応じてスケールアップまたはスケールダウンしながら、使用した分に対してのみ料金を支払うことができます。Active Directory サービスは、Windows Server 環境における ID 管理の主要なソースとして使用されます。

このセクションでは、次のトピックについて説明します。

- Amazon EC2 上のセルフマネージド Active Directory
- AWS Managed Microsoft AD
- AD Connector

Amazon EC2 上のセルフマネージド Active Directory

概要:

このセクションでは、Amazon Elastic Compute Cloud (Amazon EC2) で Active Directory を実行するコストを削減するための推奨事項を示します。主な焦点は、Active Directory ドメインコントローラーのサイズを適切に設定し、 の柔軟性を使用して環境に合わせて必要に応じて AWS クラウド 調整できるようにすることです。 AWS は、インスタンスを簡単に停止して、変化するニーズに合わせてサイズ変更したり、スケールアップが速すぎる場合はインスタンスをダウンサイズしたりできます。適切なインスタンスサイズとタイプを選択すると、大幅なコスト削減につながります。

コストへの影響

次の表は、汎用インスタンスよりもバースト可能インスタンスファミリーインスタンスを選択した場合の違いを示しています。この選択により、毎月かなりの金額を節約できます。インスタンスの適切な計画とサイズ設定は、コストの管理に役立ちます。

インスタンスタイプ	インスタンス数	vCPU	メモリ	Cost
t3a.medium	2	2	8	81.76 USD/月

インスタンスタイプ	インスタンス数	vCPU	メモリ	Cost
m5a.large	2	2	8	259.88 USD/月

コストの詳細については、 AWS 料金見積りツール 見積り を参照してください。

1 か月あたり 178.12 USD の節約は、ドメインコントローラーの年間 2,000 USD を超える節約になります。これは、1 つのアカウントで 2 つのドメインコントローラーのみの小さなフットプリント用であることに注意してください。複数のアカウントや追加のドメインコントローラーで大規模な場合、このような削減は大幅なコスト削減につながる可能性があります。

コスト最適化の推奨事項

Microsoft は、Active Directory 環境をデプロイする場合のキャパシティプランニングの推奨事項を提供します。Active Directory 環境を計画またはスケールするときは、次の主要コンポーネントを考慮することをお勧めします。

- メモリ
- Network
- Storage
- プロセッサ

これらの主要コンポーネントを念頭に置いて、 AWSの Active Directory 環境に適したインスタンスタイプを選択することができます。このセクションでは、Active Directory から AWS デプロイまでのシナリオの例をいくつか説明します。これらのシナリオでは、オンプレミス環境と同じ数のユーザーとコンピュータを処理する予定がない場合は AWS、 でオンプレミス環境をレプリケートする必要はありません。

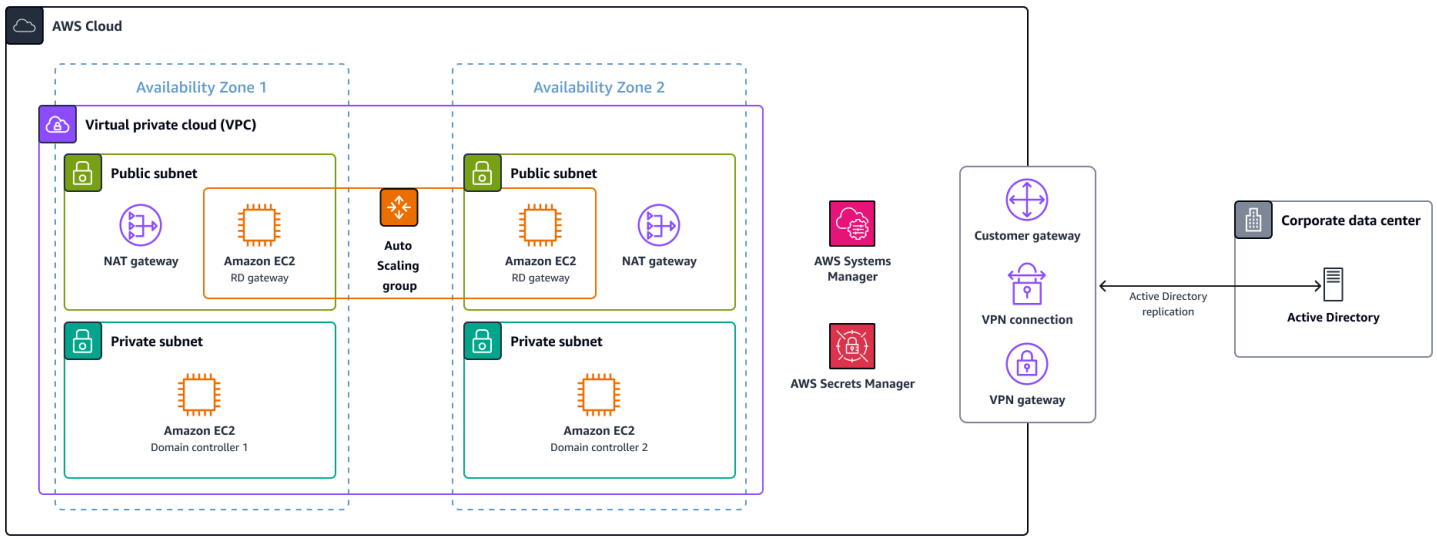
次の表は、 AWS フットプリントの vCPU、メモリ、ディスクに関する重要なコンポーネントを示しています。

コンポーネント	見積もり
ストレージ/データベースのサイズ	ユーザーごとに 40 ~ 60 KB
RAM	データセットのサイズ

コンポーネント	見積もり
	基本オペレーティングシステムの推奨事項 サードパーティーアプリケーション
Network	1 GB
CPU	コアごとに 1,000 人の同時ユーザー

ハイブリッドデプロイシナリオ

次の図は、Active Directory のハイブリッドデプロイのアーキテクチャの例を示しています。



図に示すように、通常、オンプレミスのフットプリントがあり、これを AWS クラウドに展開します。移行の初期フェーズでは、通常、すべてのユーザーとサーバーが AWSにデプロイされるわけではありません。そのため、移行作業のコストを節約するために、最初に小さなサイズのフットプリントをデプロイすることが重要です。

オンプレミスで認証するサーバーとユーザーを使用してオンプレミスフットプリントを維持する場合、 AWSのドメインコントローラーに同じフットプリントは必要ありません。Active Directory のベストプラクティスに従うことで、適切な Active Directory サイトとサービスを実装して、 AWSのドメインコントローラーへの AWS フットプリントのみを認証しながら、オンプレミスフットプリントに対してユーザーとコンピュータを認証できます。これにより、すべてのオンプレミスインフラストラクチャを使用するのではなく、 AWS リソースのみに制限 AWS することで、 での Active Directory フットプリントのオーバーサイジングを回避できます。ハイブリッドセットアップの設計

に関するガイダンスについては、Microsoft ドキュメントの「Proper placement of domain controllers and site considerations」を参照してください。

適切なサイジングによる AWS 移行の最適化

ユーザー用に Active Directory の新しいインスタンスをデプロイする場合、または Active Directory インフラストラクチャ AWS の に完全に移行する場合は、前の表で選択したインスタンスの vCPU、メモリ、ディスク容量に関する Microsoft の推奨事項に照らしてサイジングを計画することをお勧めします。

これが新しいフットプリントである場合は、小規模から始めて、簡単にインスタンスタイプを変更して AWSで拡大するにつれて環境のサイズを変更する機能を活用できます。このガイドの「Windows on Amazon EC2」セクションでは、 AWSで CPU とメモリの使用率をモニタリングおよび確認する方法について説明しています。これにより、EC2 インスタンスのサイズを増やすタイミングがわかります。

オンプレミスの Active Directory 環境を AWSに完全に移行する場合は、同じサイジングプランを実装して、適切なパフォーマンスを確保できます。オンプレミスの を複製する前に AWS、Active Directory 環境の詳細なレビューを完了することをお勧めします。これにより、オーバープロビジョニングを防ぐことができます。パフォーマンスモニターを使用して、既存のドメインコントローラーのトラフィック量と使用率に関する情報を収集するようにしてください。これにより、全体的な使用状況を理解できるため、適切なサイズを実現し、最終的にコストを削減できます。

での Active Directory の最適化 AWS

で Active Directory を実行している場合は AWS、使用量を継続的にモニタリングし、必要に応じてインスタンスサイズを変更して、支出を減らすことも重要です。を使用して AWS Compute Optimizer 、実行中のリソースに関する情報を取得できます AWS。Compute Optimizer を使用して Windows ワークロードのサイズを適正化する方法については、このガイドの「Windows on Amazon EC2」セクションを参照してください。より包括的な詳細分析のために、パフォーマンスモニター使用して Active Directory ドメインコントローラーの使用率をモニタリングし、パフォーマンスを評価して、それに応じてサイズを変更することができます。

CloudWatch を使用して、ドメインコントローラーのパフォーマンスをモニタリングすることもできます。ドメインコントローラーを最適化 (スケールアップまたはスケールダウン) するには、CloudWatch で利用可能なメトリクスを使用して、適切な意思決定を行うことができます。CloudWatch エージェントを使用して、データ収集のために送信されるカスタムパフォーマンスモニターメトリクスを設定できます。手順については、 AWS ナレッジセンターのCloudWatch エー

ジェントを使用して Windows サーバーで Performance Monitor のメトリクスを表示する方法」を参照してください。

CloudWatch エージェントをデプロイした後、`metrics_collected` のエージェント設定ファイル内で次のメトリクスを設定できます。

メトリクスカテゴリ	メトリクス名
データベースからインスタンスへ (NTDSA)	Database cache % hit
I/O database reads average latency	
I/O database reads/sec	
I/O log writes average latency	
DirectoryServices (NTDS)	LDAP bind time
DRA pending replication operations	
DRA pending replication synchronizations	
DNS	Recursive queries/sec
Recursive query failure/sec	
TCP query received/sec	
Total query received/sec	
Total response sent/sec	
UDP query received/sec	
LogicalDisk	Avg. disk queue length
% free space	
メモリ	% committed bytes in use
Long-term average standby cache lifetime(s)	

メトリクスカテゴリ	メトリクス名
ネットワークインターフェイス	Bytes sent/sec
	Bytes Received/sec
	Current bandwidth
NTDS	ATQ estimated queue delay
	ATQ request latency
	DS directory reads/sec
	DS directory searches/sec
	DS directory writes/sec
	LDAP client sessions
	LDAP searches/sec
	LDAP successful binds/sec
プロセッサ	% processor time
セキュリティシステム全体の統計	Kerberos authentications
	NTLM authentications

その他のリソース

- での Active Directory ドメインサービス AWS: パートナーソリューションデプロイガイド (AWS ドキュメント)
- 「Capacity planning for Active Directory Domain Services」(Microsoft ドキュメント)
- EC2 インスタンスで Active Directory を実行する際の設計上の考慮事項 (AWS ホワイトペーパー)

AWS Managed Microsoft AD

概要:

AWS Directory Service for Microsoft Active Directoryとも呼ばれる は AWS Managed Microsoft AD、Windows Server Active Directory を搭載し、 によって管理されます AWS。 AWS Managed Microsoft AD を使用して、さまざまな Active Directory 対応アプリケーションを に移行できます AWS クラウド。 AWS Managed Microsoft AD は、さまざまなネイティブ Active Directory アプリケーションおよびサービスと連携します。また、AWS マネージドアプリケーションおよびサービスもサポートしています。サービスとその請求メカニズム AWS Managed Microsoft AD により、 のコスト最適化レバーはあまりありませんが、コストを最小限に抑えるのに役立ついくつかの設計原則があります。

コストへの影響

AWS Managed Microsoft AD は現在の SKUs に基づくマネージドサービスであるため、サイズ設定は比較的簡単なプロセスです。現在、Standard Edition と Enterprise Edition の 2 つのサイジング SKU を使用できます。その他の SKU には、ディレクトリ共有、ドメインコントローラーの追加 (追加のリージョンを含む)、クロスリージョンデータ転送などがあります。

コスト最適化の推奨事項

AWS Managed Microsoft AD Standard Edition と AWS Managed Microsoft AD Enterprise Edition には違いがあります。Enterprise Edition は、最大 500,000 個の Active Directory オブジェクト、500 個のアカウント共有 (ソフト制限) をサポートし、マルチリージョンをサポートしています。Standard Edition は、最大 30,000 個の Active Directory オブジェクト、5 個のアカウント共有 (ソフト制限は最大約 25 個) をサポートし、マルチリージョンはサポートしていません。

> **Note**
>
> Active Directory オブジェクトの上限は近似値です。ディレクトリでサポートできるオブジェクトの数は、オブジェクトのサイズ、アプリケーションの動作やパフォーマンスニーズに応じて増減する場合があります。

ディレクトリタイプを選択する前に考慮すべき質問は次のとおりです。

- マルチリージョンのサポートは必要ですか。

- ディレクトリは 25 を超えるアカウントと共有されますか。
- Active Directory オブジェクトの数は 30,000 を超えますか。

上記の質問のいずれかに対する回答が「はい」の場合は、Enterprise Edition が必要です。すべての質問に対する回答が「いいえ」の場合は、Standard Edition から始めることをお勧めします。

> **Note**
>
> ディレクトリは Standard Edition から Enterprise Edition にアップグレードできますが、ダウングレードすることはできません。Standard Edition をデプロイすることは、一方通行の扉を通ることではありません。ディレクトリを Enterprise Edition にアップグレードする場合は、 にお問い合わせください AWS。

AWS Managed Microsoft AD Enterprise Edition でディレクトリを共有する場合は、共有ごとにコストが発生します。これは、各アカウントにディレクトリをデプロイするコストよりも安くなりますが、チェックしないと共有コストが徐々に増加する可能性があることに注意してください。Amazon Relational Database Service (Amazon RDS) および Amazon FSx for Windows File Server を含むアカウントとのみディレクトリを共有することをお勧めします。これらのサービスのみがこの機能をサポートしているためです。FSx for Windows File Server を AWS Managed Microsoft ADなどのセルフマネージド Active Directory と統合するオプションがあることに注意してください。別のアカウントで Amazon FSx のみが必要な場合は、ディレクトリを共有しなくても、 AWS Managed Microsoft AD に対してセルフマネージド Amazon FSx デプロイを実行できます。

追加のドメインコントローラーをデプロイするタイミングを決定するときは、 AWS Managed Microsoft AD が同じ VPC 内の別々のアベイラビリティーゾーンで 2 つのサブネットのみをサポートすることに注意してください。ドメインコントローラーを追加しても、サブネットを追加することはできません。パフォーマンスの問題によりドメインコントローラーを追加する必要があるかどうかを判断するには、CloudWatch でドメインコントローラーのパフォーマンスメトリクスを確認します。これにより、1 つまたはすべてのドメインコントローラーが過負荷になっているかどうかがわかります。1 つのドメインコントローラーだけが過負荷になっていると判断した場合は、ドメインコントローラーを追加しても負荷は軽減されません。現在利用可能なドメインコントローラー間で負荷分散されていないアプリケーションをさらに詳しく調べる必要があります。すべてのドメインコントローラーが頻繁に使用されている場合は、ドメインコントローラーを追加すると、既存のドメインコントローラーの負荷が軽減される可能性があります。スケーリングを自動化する方法については、 AWS セキュリティブログの「使用率メトリクスに基づいて AWS Managed Microsoft AD スケーリングを自動化する方法」を参照してください。

ディレクトリを複数のリージョンに拡張する場合は、ファイルストレージにディレクトリ NETLOGON 共有も SYSVOL 共有も使用しないことをお勧めします。すべてのドメインコントローラーは、それらの共有の内容をレプリケートします。ファイルストレージに共有を使用しないと、データ転送コストを最小限に抑えることができます。

また、 とのエンタープライズ契約に登録することもできます AWS。エンタープライズ契約では、ニーズに最適な契約を調整できます。詳細については、「エンタープライズのお客様」を参照してください。

その他のリソース

- AWS Managed Microsoft AD クォータ (AWS Directory Service ドキュメント)
- AWS Directory Service 料金表 (AWS ウェブサイト)
- 「Active Directory Domain Services on AWS」(AWS ホワイトペーパー)

AD Connector

概要:

AD Connector は、既存のオンプレミス Microsoft Active Directory を Amazon WorkSpaces、Amazon Quick、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのシームレスなドメイン結合などの互換性のあるAWS アプリケーションに、クラウドに情報をキャッシュすることなく簡単に接続できるプロキシサービスです。AD Connector を使用して、1 つのサービスアカウントを Active Directory に追加できます。AD Connector を使用すると、ディレクトリを同期化する必要がなくなり、フェデレーションインフラストラクチャをホストするコストや複雑さから解放されます。サービスとその請求メカニズムの性質上、AD Connector のコスト最適化の手段はあまりありませんが、このセクションの設計上の推奨事項に従ってコストを最小限に抑えることができます。

コストへの影響

AD Connector は、プリセット SKU に基づくマネージドサービスです。これにより、サイズ設定プロセスが簡単になります。小さいサイズと大きいサイズの 2 つのサイジング SKU を使用できます。AD Connector に関連するコスト見積もりには AWS 料金見積りツール を使用できます。

コスト最適化の推奨事項

バックエンドコンピューティングリソース以外に、小さいコネクタサイズと大きいコネクタサイズに違いはありません。

ディレクトリタイプを選択する前に考慮すべき質問は次のとおりです。

- AD Connector と統合された AWS アプリケーションを使用しているアクティブなユーザーが多数 (10,000 人以上) いますか?
- ユーザーは、多数の、深い、または循環的なネストされたグループのメンバーですか。

両方の質問に対する回答が「いいえ」の場合は、小さいサイズから始めることをお勧めします。上記の質問のいずれかに「はい」と回答した場合は、大きいサイズを検討する価値があるかもしれません。小さいサイズの AD Connector から始めて、パフォーマンスが原因でディレクトリに障害が発生した場合は、ディレクトリを大きいサイズにアップグレードするようにリクエストできます。

> **Note**
>
> AD Connector は小さいものから大きいものにアップグレードできますが、AD Connector をダウングレードすることはできません。

パフォーマンスの問題のほとんどは、AD Connector に関連するものではなく、多くのユーザーが多数の、深い、または循環的なネストされたグループのメンバーであるため過負荷になっているオンプレミスの Active Directory ドメインコントローラーに関連しています。

また、 とのエンタープライズ契約に登録することもできます AWS。エンタープライズ契約では、ニーズに最適な契約を調整できます。詳細については、「エンタープライズのお客様」を参照してください。

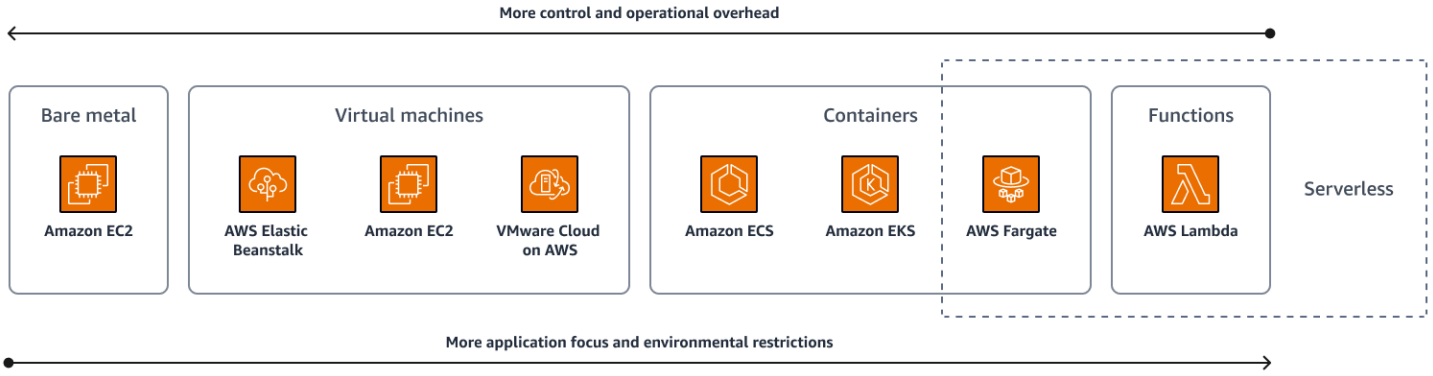
その他のリソース

- AD Connector クォータ (AWS Directory Service ドキュメント)
- その他のディレクトリタイプの料金 (AWS ウェブサイト)
- 「Active Directory Domain Services on AWS」(AWS ホワイトペーパー)

.NET

.NET アプリケーションの開発とデプロイは、クラウドコンピューティングが提供するスケールと俊敏性を実現するための重要な鍵です。多くのレガシー .NET アプリケーションの場合、 でアプリケーションを実行するのに最適なコンピューティング選択肢 AWS は、 AWS Elastic Beanstalk または Amazon Elastic Compute Cloud (Amazon EC2) のいずれかを介して仮想マシンを使用することです。Windows および Linux コンテナで .NET アプリケーションを実行することもできます。

.NET Core の導入により、すべてのクラウド上の利点を活用する最新の .NET アプリケーションを設計できます。最新のアプリケーションは、従来の一連のコンピューティング選択を使用し、 AWS Fargate や など、さまざまなタイプのサーバーレス環境をターゲットにすることもできます AWS Lambda。.NET 6+ では、Graviton2 EC2 ファミリーなどの ARM64 EC2 インスタンスでワークロードのパフォーマンスの高いホスティングが提供されるようになりました。これにより、Amazon EC2 で利用可能な最新世代のプロセッサにアクセスできます。つまり、アプリケーションは、動画エンコーディング、ウェブサーバー、ハイパフォーマンスコンピューティング (HPC) など、ワークロードタイプに特化したコンピューティングでホストできます。



このセクションでは、コスト効率を重視してクラウドの利点を活用するように .NET アプリケーションを適応させるための推奨事項を示します。

このセクションでは、次のトピックについて説明します。

- 最新の .NET にリファクタリングして Linux に移行する
- .NET アプリケーションをコンテナ化する
- Graviton インスタンスとコンテナを使用する
- 静的 .NET Framework アプリケーションの動的スケーリングをサポートする
- キャッシュを使用してデータベースの需要を減らす

- サーバーレス .NET を検討する
- 目的別データベースを検討する

最新の .NET にリファクタリングして Linux に移行する

概要:

レガシー .NET Framework アプリケーションをモダナイズすると、セキュリティ、パフォーマンス、スケーラビリティを向上させることができます。.NET Framework アプリケーションをモダナイズする効果的な方法は、最新の .NET バージョン (6+) に移行することです。これらのアプリケーションをオープンソースの .NET に移行する主な利点をいくつか紹介します。

- Linux オペレーティングシステムで実行することで Windows ライセンスコストを削減する
- 最新の言語の可用性を活用する
- Linux で実行するように最適化されたパフォーマンスを得る

多くの組織は、古いバージョンの .NET Framework をまだ実行しています。古いバージョンの脆弱性は Microsoft によって対処されなくなったため、セキュリティ上のリスクが生じる可能性があります。Microsoft は、.NET Framework の最近のバージョン (4.5.2、4.6、および 4.6.1) のサポートを終了しました。古いバージョンのフレームワークを引き続き実行する場合のリスクと利点を評価することが非常に重要です。リスクを軽減し、コストを削減するには、.NET の最新バージョンへのリファクタリングに時間と労力を費やす価値があります。

コストへの影響

コンピューティング、メモリ、ネットワークリソースのバランスを提供する汎用 EC2 インスタンスタイプ (m5) を検討してください。これらのインスタンスは、ウェブサーバー、中規模のデータベース、ソースコードリポジトリなどのさまざまなアプリケーションに適しています。

例えば、米国東部 (バージニア北部) の Windows Server (ライセンス込み) で 4 つの vCPU と 16 GB のメモリを備えたオンデマンド m5.xlarge インスタンスの料金は、月額 274.48 USD です。Linux サーバー上の同じリソースの料金は、月額 140.16 USD です。この例では、アプリケーションを .NET Framework から最新バージョンの .NET に移行し、Linux サーバーでアプリケーションを実行すると、コストが 49% 削減されます。コストは、EC2 インスタンスの選択時に指定したオプション (インスタンスタイプ、オペレーティングシステム、ストレージなど) によって異なります。Savings Plans またはリザーブドインスタンスを使用して、コストをさらに最適化できます。詳細については、AWS 料金見積りツールを使用してコスト見積もりを実行します。Windows に含まれ

るインスタンスの場合、ライセンスコストは、料金モデルに関係なく、1 時間あたり、vCPU あたり 0.046 USD です。

これらの .NET Framework アプリケーションを最新の .NET に移植するには、開発者の作業が必要です。アプリケーションとその依存関係を評価して、それらがターゲットプラットフォームバージョンと互換性があるかどうかを確認する必要があります。AWS Porting Assistant for .NET は、.NET Framework アプリケーションをスキャンして .NET 互換性評価を生成する支援ツールであり、アプリケーションをより迅速に Linux 互換へ移植できるように支援します。Porting Assistant for .NET は、.NET との非互換性を特定し、既知の代替品を見つけて、詳細な互換性評価を生成します。ソリューションを移植した後、依存関係を使用してプロジェクトを正常にコンパイルするには、手動でコードを変更する必要があります。これにより、アプリケーションの Linux へのモダナイズに伴う手動作業が軽減されます。アプリケーションが ARM プロセッサをサポートしている場合、Linux に移行すると、Graviton インスタンスを使用する機能がロック解除されます。これにより、さらに 20% のコスト削減を実現できます。詳細については、 AWS コンピューティングブログの「Powering .NET 5 with AWS Graviton2: Benchmarks」を参照してください。

AWS Toolkit for .NET Refactoring や .NET Upgrade Assistant など、レガシー .NET Framework アプリケーションを最新の .NET に移行するのに役立つツールもあります。

コスト最適化の推奨事項

.NET Framework アプリケーションを移行するには、以下を実行します。

1. 前提条件 – Porting Assistant for .NET を使用するには、アプリケーションのソースコードを分析するマシンに .NET 5+ をインストールする必要があります。マシンのリソースとして、1.8 GHz 以上の処理速度、4 GB のメモリ、5 GB のストレージ容量が必要です。詳細については、Porting Assistant for .NET ドキュメントの「Prerequisites」を参照してください。
2. 評価 – Porting Assistant for .NET を実行可能 (ダウンロード) ファイルとしてダウンロードします。ツールをダウンロードしてマシンにインストールし、アプリケーションの評価を開始できます。評価ページには、最新の .NET と互換性のない移植されたプロジェクト、パッケージ、API が含まれています。このため、評価後にソリューションでビルドエラーが発生します。評価結果を表示または CSV ファイルにダウンロードできます。詳細については、Porting Assistant for .NET ドキュメントの「Port a solution」を参照してください。
3. リファクタリング — アプリケーションを評価した後、プロジェクトをターゲットフレームワークバージョンに移植できます。ソリューションを移植すると、プロジェクトファイルと一部のコードが Porting Assistant によって変更されます。ログをチェックして、ソースコードの変更を確認できます。ほとんどの場合、コードは、移行とテストを完了して本番環境の準備を整えるために、追加の作業が必要になります。アプリケーションによっては、エンティティフレームワー

ク、アイデンティティ、認証などの変更が含まれる場合があります。詳細については、Porting Assistant for .NET ドキュメントの「Port a solution」を参照してください。

これは、アプリケーションをコンテナにモダナイズするための最初のステップです。.NET Framework アプリケーションを Linux コンテナにモダナイズするビジネス上および技術上の推進要因は数多く考えられます。重要な推進要因の 1 つは、Windows オペレーティングシステムから Linux に移行することで、総保有コストを削減することです。これにより、アプリケーションをクロスプラットフォームバージョンの .NET およびコンテナに移行してリソース使用率を最適化する際のライセンスコストを削減できます。

アプリケーションを Linux に移植したら、AWS App2Container を使用してアプリケーションをコンテナ化できます。App2Container は、直接デプロイできるエンドポイントサービスとして Amazon ECS または Amazon EKS を使用します。App2Container は、アプリケーションを繰り返しコンテナ化するために必要なすべての Infrastructure as Code (IaC) デプロイアーティファクトを提供します。

その他の考慮事項とリソース

- (2002 年のレガシーフレームワーク) 上に構築されたアプリケーションがあり、それらを .NET 6 に移植する場合は、 AWS ブログの「Porting Assistant for .NET を使用してレガシー VB.NET アプリケーションを .NET 6.0 に移植する」投稿を参照してください。
- Windows Communication Foundation (WCF) にレガシーアプリケーションがあり、最新の .NET で実行する場合は、CoreWCF を採用できます。詳細については、 AWS ブログの Microsoft Workloads の「Porting Assistant for .NET を使用した CoreWCF へのレガシー WCF アプリケーションのモダナイズ」を参照してください。
- 移植アシスタントを Visual Studio IDE の拡張機能として追加できます。これにより、IDE と Porting Assistant for .NET ツールを切り替えることなく、コードの変換に必要なすべてのタスクを実行できます。詳細については、 AWS ブログの Microsoft Workloads の「Accelerate .NET application modernization with Porting Assistant for .NET Visual Studio IDE extension post」を参照してください。
- AWS Porting Assistant for .NET は、評価のソースコードと互換性分析コンポーネントを備えたオープンソースツールになりました。これにより、開発者が .NET 移植の知識とベストプラクティスを使用し、共有することが促進されます。
- AWS Toolkit for .NET リファクタリングを使用して、.NET Framework アプリケーションを最新の .NET on Linux に移植できます。詳細については、ブログの Microsoft Workloads AWS の「Accelerate .NET modernization with AWS Toolkit for .NET Refactoring」の投稿を参照してください。

- ASP.NET Core アプリケーションのコンテナ化と AWS を使用した への移行を加速 AWS App2Containerできます。

.NET アプリケーションをコンテナ化する

概要:

コンテナは、一貫して再現可能な方法でアプリケーションをパッケージ化およびデプロイするための軽量で効率的な方法です。このセクションでは、サーバーレスコンテナサービスである AWS Fargateを使用して .NET アプリケーションのコストを削減し、スケーラブルで信頼性の高いインフラストラクチャを提供する方法について説明します。

コストへの影響

コスト削減のためのコンテナの使用の有効性に影響を与える要因には、アプリケーションのサイズと複雑さ、デプロイする必要があるアプリケーションの数、アプリケーションのトラフィックと需要のレベルなどがあります。小規模または単純なアプリケーションの場合、コンテナと関連サービスの管理のオーバーヘッドによって実際にコストが増加する可能性があるため、従来のインフラストラクチャアプローチと比較して、コンテナでは大幅なコスト削減が得られない可能性があります。ただし、大規模または複雑なアプリケーションの場合、コンテナを使用すると、リソース使用率を向上させ、必要なインスタンスの数を減らすことでコスト削減を実現できます。

コスト削減のためにコンテナを使用する場合は、次の点に留意することをお勧めします。

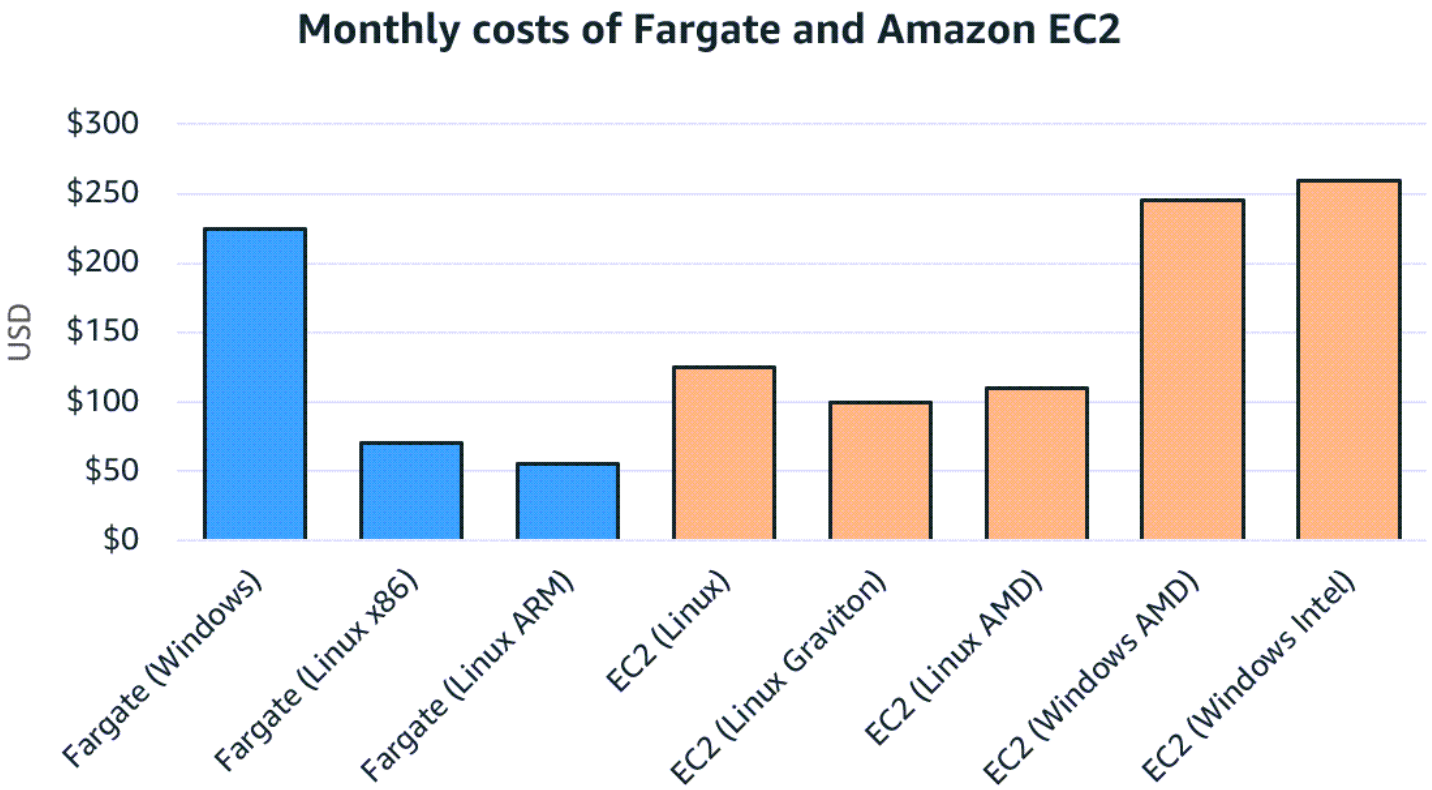
- アプリケーションのサイズと複雑さ – より大規模で複雑なアプリケーションは、より多くのリソースを必要とする傾向があり、リソース使用率の向上からより多くのメリットが得られるため、コンテナ化に適しています。
- アプリケーションの数 – 組織がデプロイする必要があるアプリケーションの数が増えるほど、コンテナ化によって実現できるコスト削減が大きくなります。
- トラフィックと需要 – トラフィックと需要が多いアプリケーションは、コンテナが提供するスケーラビリティと伸縮性の恩恵を受けることができます。これにより、コスト削減につながる可能性があります。

アーキテクチャやオペレーティングシステムが異なると、コンテナのコストに影響します。Windows コンテナを使用している場合、ライセンス上の考慮事項により、コストが削減されない場合があります。Linux コンテナでは、ライセンスコストが低くなるか、ライセンスコストがかか

りません。次のグラフでは、米国東部 (オハイオ) リージョン AWS Fargate の の基本設定と、4 つの vCPUs と 8 GB のメモリが割り当てられた 12 時間実行される 1 か月あたり 30 個のタスクを使用します。

EC2 ベースのコンテナホストとサーバーレスまたは AWSの 2 つのプライマリコンピューティングプラットフォームからコンテナを実行できますAWS Fargate。 EC2-based Fargate の代わりに Amazon Elastic Container Service (Amazon ECS) を使用する場合は、実行中のコンピューティング (インスタンス) を維持し、必要に応じてプレイスメントエンジンがコンテナをインスタンス化できるようにする必要があります。代わりに Fargate を使用する場合、必要なコンピューティングキャパシティのみがプロビジョニングされます。

次のグラフは、Fargate を使用した場合と Amazon EC2 を使用した場合の同等のコンテナの違いを示しています。Fargate の柔軟性により、アプリケーションのタスクは 1 日 12 時間実行でき、オフ時間には使用率をゼロにすることができます。ただし、Amazon ECS の場合は、EC2 インスタンスの Auto Scaling グループを使用してコンピューティングキャパシティを制御する必要があります。これにより、キャパシティが 1 日 24 時間稼働することになり、最終的にコストが増加する可能性があります。



コスト最適化の推奨事項

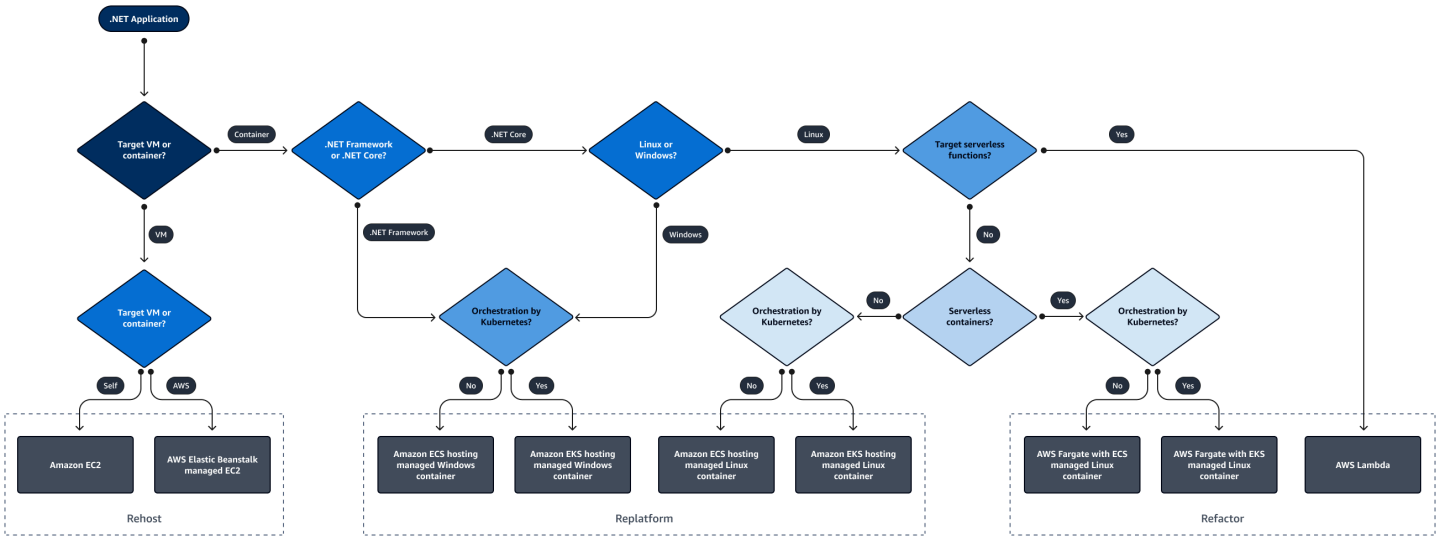
Windows ではなく Linux コンテナを使用する

Windows コンテナの代わりに Linux コンテナを使用すると、大幅なコスト削減を実現できます。例えば、EC2 Windows で .NET Framework を実行するのではなく、EC2 Linux で .NET Core を実行すると、コンピューティングコストを約 45% 削減できます。x86 の代わりに ARM アーキテクチャ (AWS Graviton) を使用すると、さらに 40% の割引を受けることができます。

既存の .NET Framework アプリケーション用に Linux ベースのコンテナを実行する場合は、Linux コンテナを使用するために、これらのアプリケーションを最新のクロスプラットフォームバージョンの .NET (.NET 6.0 など) に移植する必要があります。重要な考慮事項は、リファクタリングのコストと、Linux コンテナのコスト低下によって得られるコスト削減を比較検討することです。最新の .NET へのアプリケーションの移植の詳細については、 AWS ドキュメントの「Porting Assistant for .NET」を参照してください。

最新の .NET に移行する (つまり、.NET Framework から移行する) もう 1 つの利点は、追加のモダナイズの機会が利用可能になることです。例えば、アプリケーションを、よりスケーラブルで俊敏で、コスト効率の高いマイクロサービスベースのアーキテクチャにリアーキテクトすることを検討できます。

次の図は、モダナイズの機会を探るための意思決定プロセスを示しています。



Savings Plans を利用する

コンテナを使用すると、Compute Savings Plans を活用して Fargate のコストを削減できます。柔軟な割引モデルは、コンバーティブルリザーブドインスタンスと同じ割引を提供します。Fargate の

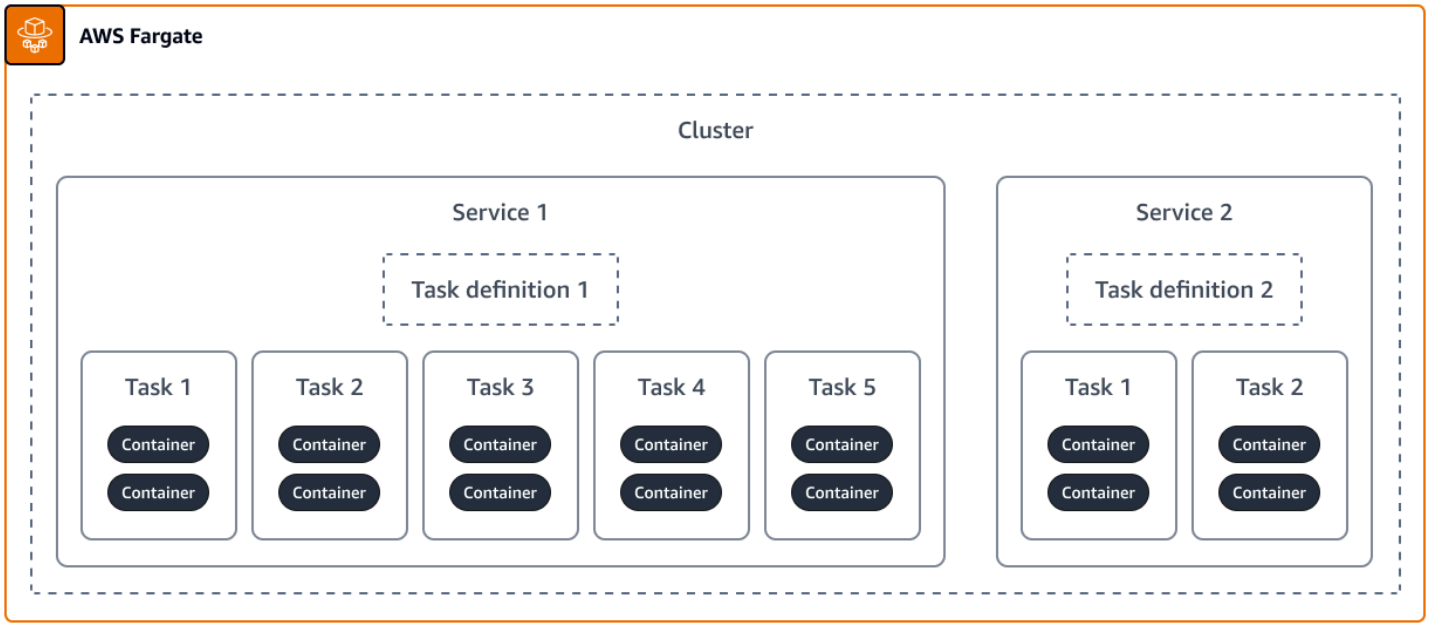
料金は、コンテナイメージのダウンロードを開始した時点から Amazon ECS タスクが終了するまで (最も近い秒に切り上げ) に使用された vCPU およびメモリリソースに基づいています。Fargate の Savings Plans は、1 年または 3 年の期間に特定の量のコンピューティング使用量 (1 時間あたりのドルで測定) を使用するコミットメントと引き換えに、Fargate の使用量に対して最大 50% のコスト削減を提供します。AWS Cost Explorer を使用して Savings Plans を選択できます。

Compute Savings Plans は、最も大きなコスト削減を最初に得られる使用量に適用されることを理解することが重要です。例えば、`us-east-2` で t3.medium Linux インスタンスを実行し、同一の Windows t3.medium インスタンスを実行している場合、Linux インスタンスが最初に Savings Plans の特典を受け取ります。これは、Linux インスタンスのコスト削減の可能性は 50% であるのに対し、同じ Windows インスタンスのコスト削減の可能性は 35% であるためです。Amazon EC2 や Lambda など AWS アカウント、他の Savings Plans 対象リソースが で実行されている場合、Savings Plans を最初に Fargate に適用する必要はありません。詳細については、Savings Plans ドキュメント」の「Savings Plans AWS が使用状況にどのように適用されるか を理解する」と、このガイドのAmazon EC2 での Windows の支出を最適化する」セクションを参照してください。
Savings Plans

Fargate タスクのサイズを適切に設定する

最大限のコスト最適化を実現するには、Fargate タスクのサイズが正しく設定されていることを確認することが重要です。多くの場合、開発者は、アプリケーションで使用される Fargate タスクの設定を最初に決定する際に、必要なすべての使用状況情報を持っているわけではありません。これにより、タスクの過剰プロビジョニングが発生し、不要な支出が発生する可能性があります。これを回避するには、Fargate で実行されているテストアプリケーションをロードして、さまざまな使用シナリオで特定のタスク設定がどのように実行されるかを理解することをお勧めします。負荷テスト結果、vCPU、タスクのメモリ割り当て、自動スケーリングポリシーを使用して、パフォーマンスとコストの適切なバランスを見つけることができます。

次の図は、Compute Optimizer が最適なタスクおよびコンテナサイズの推奨事項を生成する方法を示しています。



1 つのアプローチは、分散負荷テストで説明されているような負荷テスト AWSツールを使用して、vCPU とメモリ使用率のベースラインを確立することです。負荷テストを実行して一般的なアプリケーションの負荷をシミュレートした後、ベースライン使用率が達成されるまでタスクの vCPU とメモリの設定をファインチューニングできます。

その他のリソース

- 「Cost Optimization Checklist for Amazon ECS and AWS Fargate」(AWS Containers ブログ記事)
- Amazon ECS 起動タイプ別の理論コスト最適化: Fargate vs EC2 (AWS コンテナブログ記事)
- Porting Assistant for .NET (AWS ドキュメント)
- での分散負荷テスト AWS (AWS ソリューションライブラリ)
- AWS Compute Optimizer が で Amazon ECS サービスのサポートを開始 AWS Fargate (AWS Cloud Financial Management ブログ記事)

Graviton インスタンスとコンテナを使用する

概要:

AWS Graviton インスタンスは、Amazon Elastic Compute Cloud (Amazon EC2) で実行されているコンテナを含め、Amazon Elastic Compute Cloud (Amazon EC2) で実行されているクラウドワークロードに最適な価格パフォーマンスを提供する AWS ように によって設計された ARM プロセッサを搭載しています AWS。現在、Amazon EC2 で使用できる Graviton は 3 世代あります。この

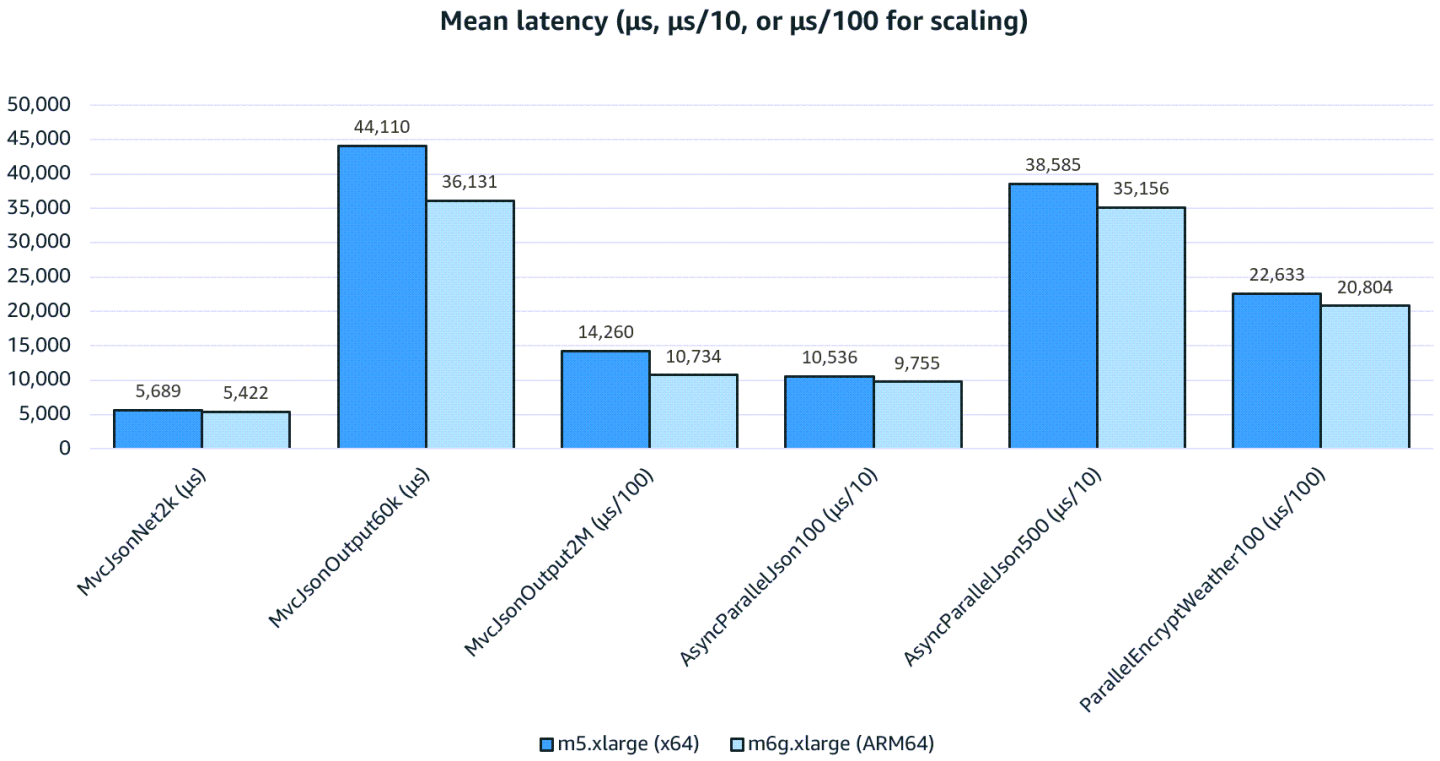
ガイドでは、.NET アプリケーションでの Graviton 2 および 3 の使用に焦点を当てています。これは、Graviton の最新バージョンを使用すると大幅なコスト削減になるためです。Graviton インスタンスは Linux オペレーティングシステムのみを実行することに注意してください。結果として、Graviton インスタンスは Linux で実行されている .NET の強力なサービスですが、Windows オペレーティングシステムやレガシー .NET Framework アプリケーションのオプションではありません。

Graviton 3 は、同等の EC2 インスタンスよりも 60% 効率的であり、パフォーマンスは最大 40% 向上します。このガイドでは、Graviton を使用するコスト上の利点に焦点を当てていますが、Graviton にはパフォーマンスの向上と環境の持続可能性の向上という追加の利点があることに注意してください。

コストへの影響

Graviton に切り替えると、最大 45% のコスト削減を実現できます。レガシー .NET Framework アプリケーションを最新の .NET バージョンにリファクタリングしたら、Graviton インスタンスを使用する機能をロック解除します。Graviton への移行は、.NET 開発者にとって効果的なコスト最適化手法です。

次の表の例では、Graviton インスタンスに移行することで達成できるパフォーマンス向上の可能性を示しています。



前述の図の結果の作成に使用されるベンチマークアプローチの完全な内訳と説明については、 AWS コンピューティングブログの「Powering .NET 5 with AWS Graviton2: Benchmarks」を参照してください。

効率向上の理由の 1 つは、x86 と Graviton における vCPU の意味の違いです。x86 アーキテクチャでは、vCPU はハイパースレッディングによって達成される論理コアです。Graviton では、vCPU は物理コアに相当し、vCPU をワークロードに完全にコミットできます。

Graviton2 での結果は、同等の x86/x64 インスタンスよりも 40% 優れた価格パフォーマンスを示しています。Graviton2 に対して、Graviton3 は以下を提供します。

- パフォーマンスプロファイルが強化され、パフォーマンスが 25% 向上
- 浮動小数点のパフォーマンスが最大 2 倍向上
- 暗号化ワークロードのパフォーマンスが最大 2 倍高速化
- 機械学習のパフォーマンスが最大 3 倍向上

さらに、Graviton3 は、DDR5 メモリを搭載したクラウド上の最初のインスタンスです。

次の表は、Graviton ベースのインスタンスと、同等の x86 ベースのインスタンスとのコスト削減の違いを示しています。

この表は、Graviton による 19.20% のコスト削減を示しています。

インスタンスタイプ	アーキテクチャ	vCPU	メモリ (GB)	時間単位のコスト (オンデマンド)
t4g.xlarge	ARM	4	16	0.1344 USD
t3.xlarge	x86	4	16	0.1664 USD

この表は、Graviton による 14.99% のコスト削減を示しています。

インスタンスタイプ	アーキテクチャ	vCPU	メモリ (GB)	時間単位のコスト (オンデマンド)
c7g.4xlarge	ARM	16	32	0.5781 USD
c6i.4xlarge	x86	16	32	0.6800 USD

Graviton を検討するときは、アプリケーションのパフォーマンスプロファイルをテストすることが重要です。Graviton は、強固なソフトウェア開発プラクティスに代わるものではありません。テストを使用して、基盤となるコンピューティングリソースを最大限に活用しているかどうかを確認できます。

コスト最適化の推奨事項

Graviton プロセッサ/インスタンスを利用する方法はいくつかあります。このセクションでは、x86 アーキテクチャマシンの使用から Graviton (ARM) インスタンスへの移行に必要な変更点について説明します。

Lambda でランタイム設定を変更する

ランタイム設定を切り替えることをお勧めします AWS Lambda。詳細については、Lambda ドキュメントの「ランタイム環境の変更」を参照してください。.NET はコンパイルされた言語であるため、これを機能させるにはビルドプロセスに従う必要があります。これを行う方法の例については、GitHub の「.NET on Graviton」を参照してください。

コンテナ

コンテナ化されたワークロードの場合は、マルチアーキテクチャコンテナイメージを作成します。これを行うには、Docker ビルドコマンドで複数のアーキテクチャを指定します。例えば、次のようになります。

```
docker buildx build -t "myImageName:latest" --platform linux/amd64,linux/arm64 --push .
```

などのツールを使用してビルドを AWS Cloud Development Kit (AWS CDK) オーケストレーションすることもできます。 https://aws.amazon.com/blogs/devops/build-and-deploy-net-web-applications-to-arm-powered-aws-graviton-2-amazon-ecs-clusters-using-aws-cdk/Docker の例については、Docker

ドキュメントの「Building Multi-Arch Images for Arm and x86 with Docker Desktops」を参照してください。

Amazon EC2

x86/x64 から ARM に移行するには、コンパイルステップで ARM アーキテクチャをターゲットにします。Visual Studio では、ARM64 CPU を作成できます。手順については、Microsoft ドキュメントの「To configure a project to target Arm64 and other platforms」を参照してください。

.NET CLI を使用している場合、ARM マシンでビルドを実行すると、Graviton 互換ビルドが生成されます。デモを確認するには、YouTube で Accelerate .NET 6 performance with Arm64 on AWS Graviton2 をご覧ください。依存関係の問題により、コンパイル時エラーが発生しますが、個別に対処できます。依存関係に対して ARM ライブラリが存在する限り、移行は比較的容易であるはずです。

その他のリソース

- ARM 用のコンテナを構築し、Amazon ECS の Graviton インスタンスとスポットインスタンスで保存する方法 (AWS ブログ)
- AWS LambdaAWS Graviton2 プロセッサを搭載した関数 – 関数をアームで実行し、最大 34% の価格パフォーマンスを向上 (AWS ブログ)
- Arm-based AWS Graviton2 プロセッサへの AWS Lambda 関数の移行 (AWS ブログ)
- を使用して .NET ウェブアプリケーションを構築して ARM 搭載 AWS の Graviton 2 Amazon ECS クラスターにデプロイする AWS CDK (AWS ブログ)
- Graviton Fast Start – ワークロードを Graviton AWS に移動するのに役立つ新しいプログラム (AWS ブログ)
- AWS Graviton2 を使用した .NET 5 の強化: ベンチマーク (AWS ブログ)

静的 .NET Framework アプリケーションの動的スケーリングをサポートする

概要:

アプリケーションにクラウドを使用する主な利点の 1 つは、伸縮性、つまり需要に基づいてコンピューティングをスケールインまたはスケールアウトする機能です。これにより、ピーク時の使用量に合わせてプロビジョニングするのではなく、必要なコンピューティングキャパシティに対してのみ

支払うことができます。オンライン小売業者が通常の何倍ものトラフィック (例えば、数分以内に数千パーセント) をすぐに取得できるサイバーマンデーは、伸縮性の良い例です。

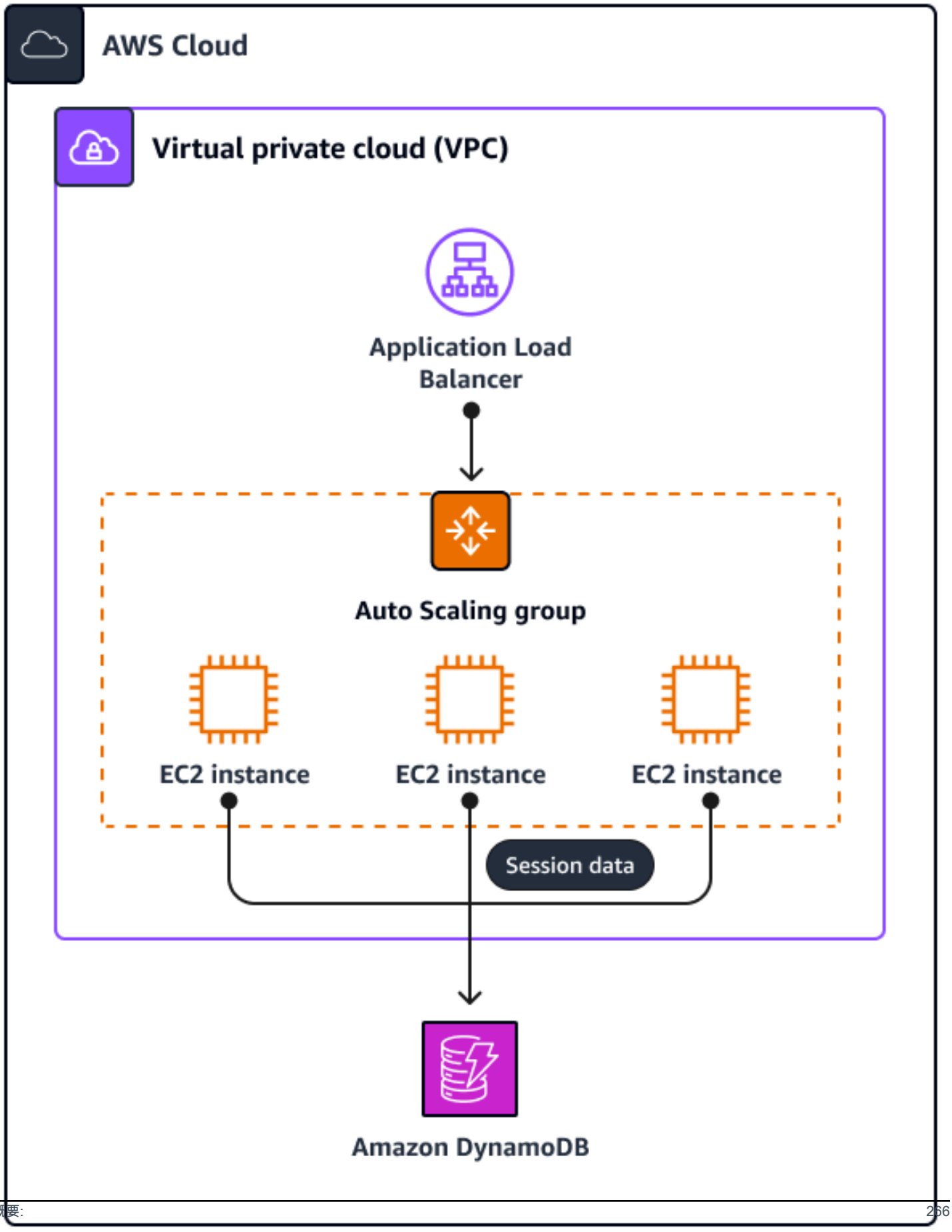
レガシー .NET ウェブアプリケーション (IIS で実行されている ASP.NET Framework アプリケーションなど) をクラウドに持ち込む場合、アプリケーションのステートフルな性質上、負荷分散されたサーバーファームを迅速にスケーリングすることは困難または不可能な場合があります。ユーザーセッションデータは、通常、ASP.NET セッション状態または持続する必要がある相互要求データを保持する静的変数を使用して、アプリケーションのメモリ内に保存されます。ユーザーセッションのアフィニティは、通常、ロードバランサーのスティッキーセッションを通じて維持されます。

これは運用上困難であることが判明しています。キャパシティを増やす必要がある場合は、意図的にサーバーをプロビジョニングして追加する必要があります。これは時間のかかるプロセスになる場合があります。パッチ適用や予期しない障害が発生した場合にノードをサービス停止にすると、エンドユーザーエクスペリエンスに問題が発生し、影響を受けるノードに関連付けられているすべてのユーザーの状態が失われる可能性があります。これにより、うまくいっても、ユーザーは再度ログインする必要が出てきます。

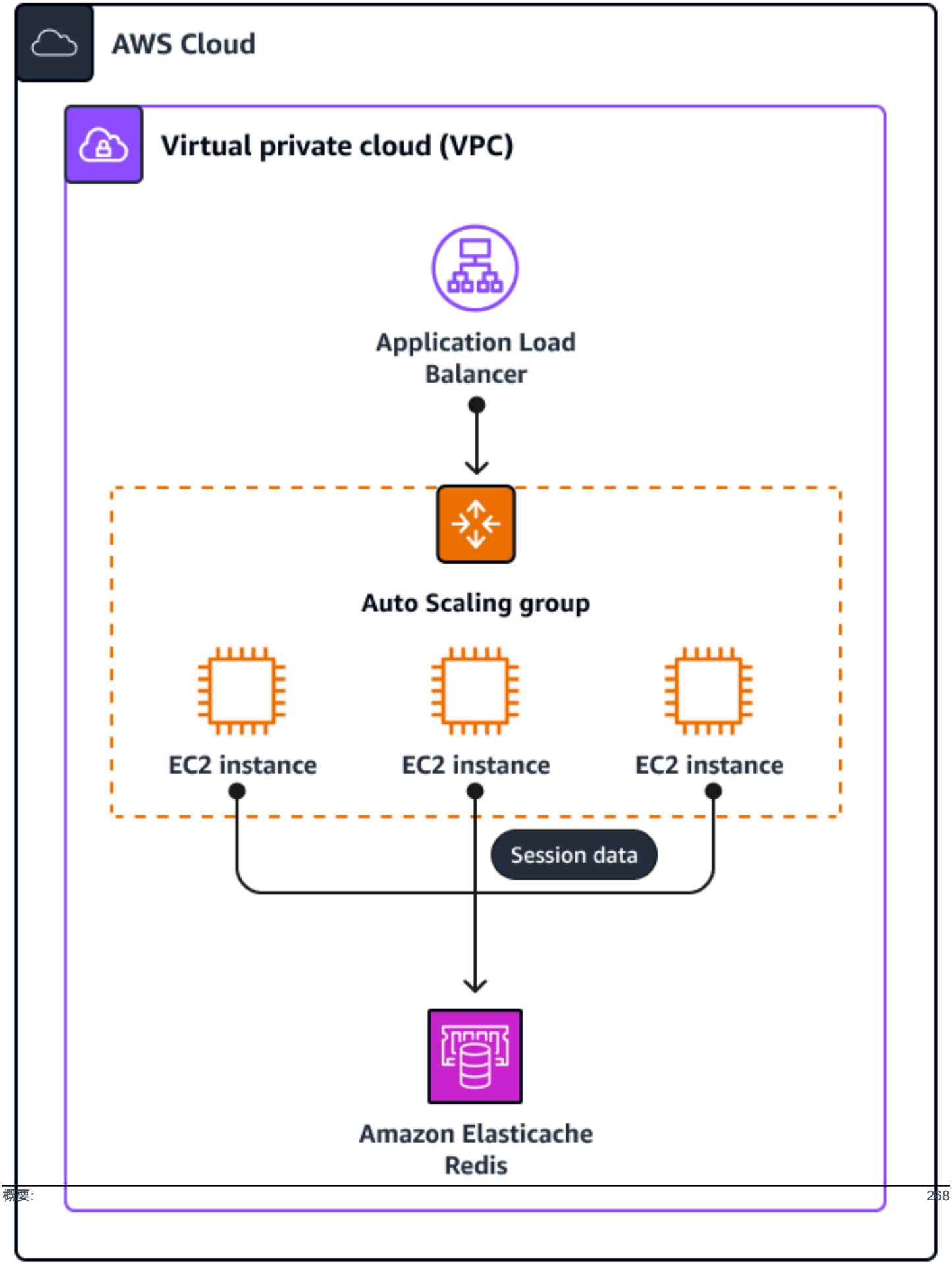
ASP.NET アプリケーションのセッション状態を一元化し、Auto Scaling ルールをレガシー ASP.NET アプリケーションに適用することで、クラウドの伸縮性を活用でき、場合によってはアプリケーションの実行時にコスト削減を活用できます。例えば、コンピューティングのスケーラビリティによりコストを削減できますが、リザーブドインスタンスの使用量の削減や Amazon スポットインスタンスの料金表の使用など、利用可能なさまざまな料金モデルから選択することもできます。

2 つの一般的な手法には、セッション状態プロバイダーとして Amazon DynamoDB を使用することと、ASP.NET セッションストアとして Amazon ElastiCache (Redis OSS) を使用することが含まれます。

次の図は、セッション状態プロバイダーとして DynamoDB を使用するアーキテクチャを示しています。


AWS Cloud
Virtual private cloud (VPC)
Application Load Balancer
Auto Scaling group
EC2 instance
EC2 instance
EC2 instance
Session data
Amazon DynamoDB

次の図は、セッション状態プロバイダーとして ElastiCache (Redis OSS) を使用するアーキテクチャを示しています。

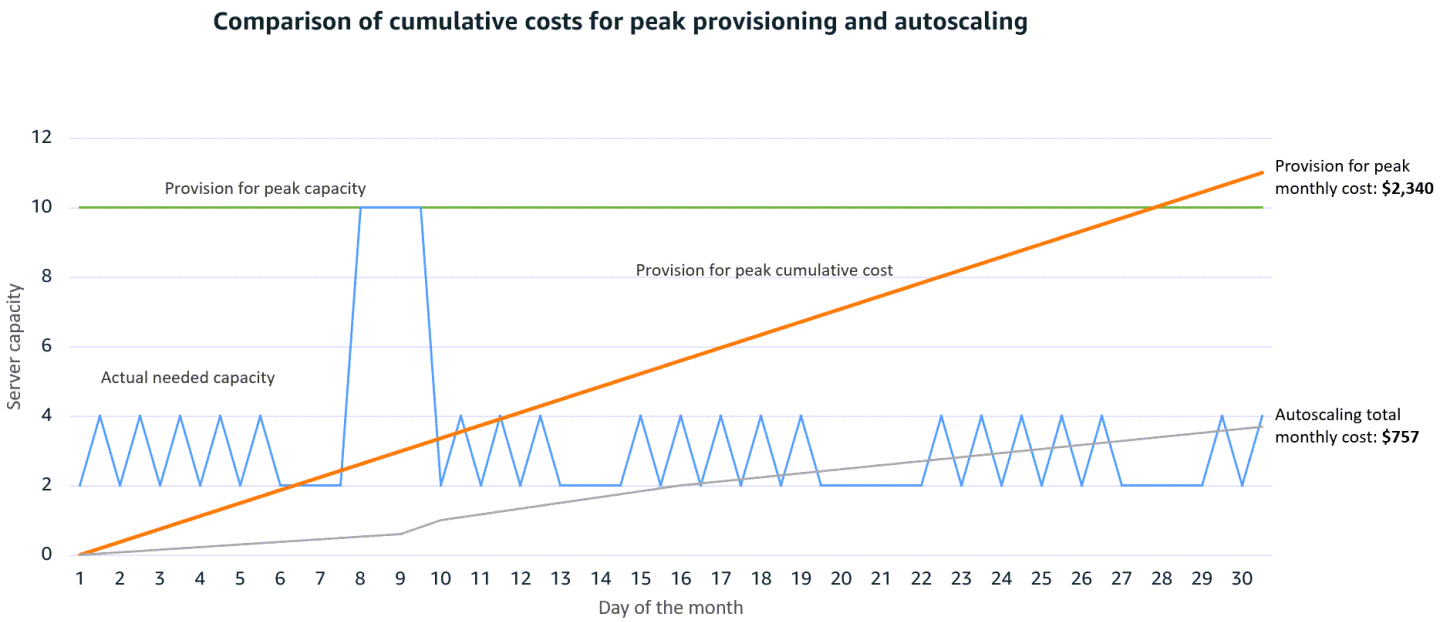

AWS Cloud
Virtual private cloud (VPC)
Application Load Balancer
Auto Scaling group
EC2 instance
EC2 instance
EC2 instance
Session data
Amazon Elasticache Redis

コストへの影響

本番アプリケーションのスケーリングの利点を判断するには、実際の需要をモデル化することをお勧めします。このセクションでは、サンプルアプリケーションをモデル化するために次のように仮定します。

- ローテーションに追加および削除されるインスタンスは同一であり、インスタンスサイズのバリエーションは導入されません。
- アプリケーションの高可用性を維持するために、サーバー使用率が 2 つのアクティブサーバーを下回ることはありません。
- サーバーの数は、トラフィックに合わせて直線的にスケールされます (つまり、トラフィックが 2 倍になると、コンピューティングも 2 倍必要になります)。
- トラフィックは 1 か月にわたって 6 時間単位でモデル化され、1 日内の変動と、1 日の 10 倍のトラフィックという 1 回の異常なトラフィックピーク (プロモーションセールなど) が含まれます。週末トラフィックは、基本使用率に基づいてモデル化されます。
- 夜間トラフィックは基本使用率でモデル化され、平日トラフィックは 4 倍の使用率でモデル化されます。
- リザーブドインスタンスの料金では、1 年間の前払いなしの料金が使用されます。通常の日中の料金ではオンデマンド料金を使用し、バースト需要ではスポットインスタンス料金を使用します。

次の図は、このモデルがピーク時の使用のためにプロビジョニングするのではなく、.NET アプリケーションで伸縮性を活用する方法を示しています。これにより、約 68% のコスト削減になります。



セッション状態ストレージメカニズムとして DynamoDB を使用する場合は、次のパラメータを使用します。

```
Storage: 20GB
Session Reads: 40 million
Session Writes: 20 million
Pricing Model: On demand
```

このサービスの推定月額コストは、1 か月あたり約 35.00 USD です。

セッション状態ストレージメカニズムとして ElastiCache (Redis OSS) を使用する場合は、次のパラメータを使用します。

```
Number of Nodes: 3
Node size: cache.t4g.medium
Pricing Model: 1y reserved
```

このサービスの推定月額コストは、1 か月あたり約 91.00 USD です。

コスト最適化の推奨事項

最初のステップは、レガシー .NET アプリケーションにセッション状態を実装することです。ステートストレージメカニズムとして ElastiCache を使用している場合は、 AWS デベロッパーツールブログの ASP.NET セッションストアとして ElastiCache のガイダンスに従ってください。DynamoDB

を使用している場合は、 SDK for .NET ドキュメントの What is the AWS SDK for .NETのガイダンスに従ってください。

アプリケーションが最初に InProc セッションを使用する場合は、セッションに保存する予定のすべてのオブジェクトをシリアル化できることを確認してください。これを行うには、SerializableAttribute 属性を使用して、インスタンスがセッションに保存されるクラスを修飾します。例えば、次のようになります。

```
[Serializable()]
public class TestSimpleObject {
public string SessionProperty {get;set;}
}
```

さらに、.NET MachineKey は使用中のすべてのサーバー間で同じである必要があります。これは通常、インスタンスが共通の Amazon マシンイメージ (AMI) から作成された場合です。例えば、次のようになります。

```
<machineKey
validationKey="some long hashed value"
decryptionKey="another long hashed value"
validation="SHA1"/>
```

ただし、ベースイメージが変更された場合は、同じ .NET マシンイメージ (IIS またはサーバーレベルで設定可能) で設定されていることを確認することが重要です。詳細については、Microsoft ドキュメントの「SystemWebSectionGroup.MachineKey Property」を参照してください。

最後に、スケーリングイベントに応じて Auto Scaling グループにサーバーを追加するメカニズムを決定する必要があります。これを行うには、いくつかの方法があります。Auto Scaling グループの EC2 インスタンスに .NET Framework アプリケーションをシームレスにデプロイするには、次の方法をお勧めします。

- EC2 Image Builder を使用して、完全に設定されたサーバーとアプリケーションを含む AMI を設定します。その後、この AMI を使用して、Auto Scaling グループの起動テンプレートを設定できます。
- AWS CodeDeploy を使用してアプリケーションをデプロイします。CodeDeploy では、Amazon EC2 Auto Scaling と直接統合できます。これにより、アプリケーションのリリースごとに新しい AMI を作成する代わりに別の方法が提供されます。

その他のリソース

- 「Create images with EC2 Image Builder」(EC2 Image Builder ドキュメント)
- Visual Studio Team Services AWS CodeDeploy を使用した .NET ウェブアプリケーションのデプロイ (AWS 開発者ツールブログ)

キャッシュを使用してデータベースの需要を減らす

概要:

キャッシュを効果的な戦略として使用し、.NET アプリケーションのコストを削減できます。アプリケーションがデータへの頻繁なアクセスを必要とする場合、多くのアプリケーションは、SQL Server などのバックエンドデータベースを使用します。需要に対応するためにこれらのバックエンドサービスを維持するコストは高くなる可能性がありますが、効果的なキャッシュ戦略を使用して、サイジングとスケーリングの要件を減らすことでバックエンドデータベースの負荷を軽減できます。これにより、コストを削減し、アプリケーションのパフォーマンスを向上させることができます。

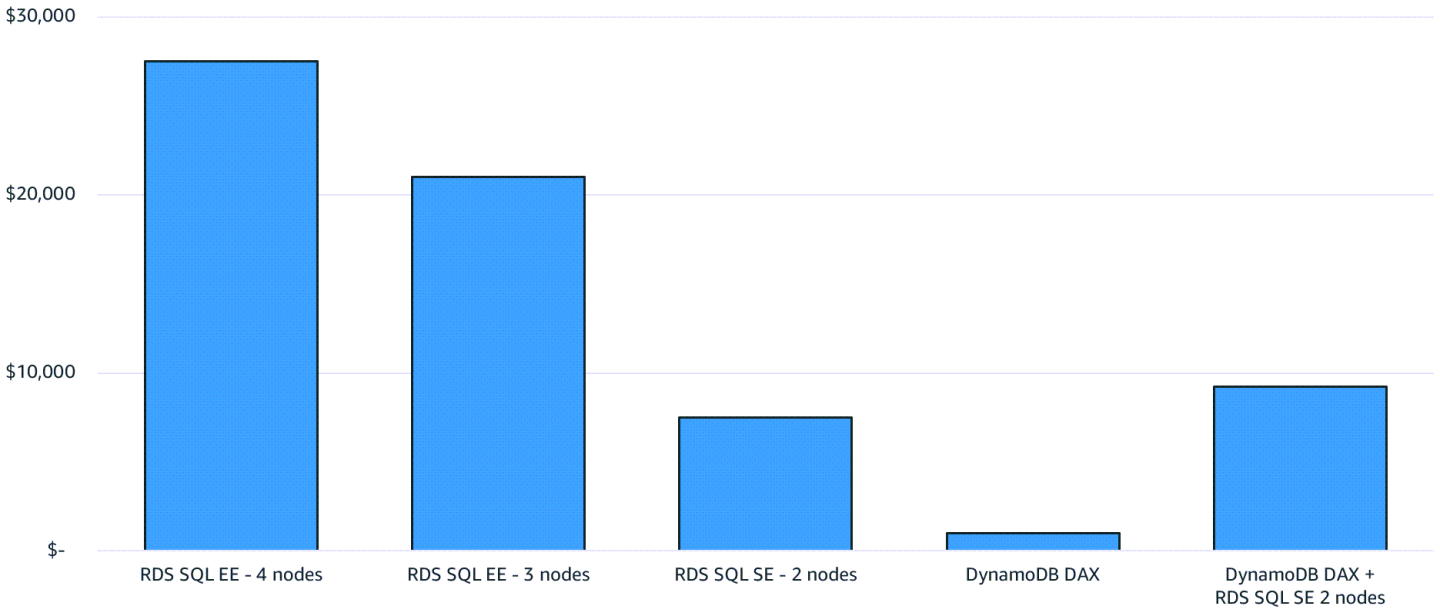
キャッシュは、SQL Server などのより高価なリソースを使用する読み込み負荷の高いワークロードに関連するコストを節約するのに役立つ手法です。ワークロードに適した手法を使用することが重要です。例えば、ローカルキャッシュはスケーラブルではなく、アプリケーションのインスタンスごとにローカルキャッシュを維持する必要があります。基盤となるデータソースのコストが低いほどキャッシュメカニズムに関連する追加コストが相殺されるように、パフォーマンスへの影響を潜在的なコストと比較検討する必要があります。

コストへの影響

SQL Server では、データベースのサイズ設定時に読み取りリクエストを考慮する必要があります。これは、負荷に対応するためにリードレプリカを導入する必要がある場合があるため、コストに影響する可能性があります。リードレプリカを使用する場合は、SQL Server Enterprise Edition でのみ使用できることを理解することが重要です。このエディションには、SQL Server Standard Edition よりも高価なライセンスが必要です。

次の図は、キャッシュの有効性を理解するのに役立つように設計されています。SQL Server Enterprise Edition を実行している 4 つの db.m4.2xlarge ノードを持つ Amazon RDS for SQL Server を示しています。これは、1 つのリードレプリカを持つマルチ AZ 設定でデプロイされます。排他的な読み取りトラフィック (SELECT クエリなど) は、リードレプリカに送信されます。これに対して、Amazon DynamoDB は r4.2xlarge の 2 ノード DynamoDB Accelerator (DAX) クラスターを使用します。

次のグラフは、多くの読み取りトラフィックを処理する専用リードレプリカの必要性をなくした結果を示しています。



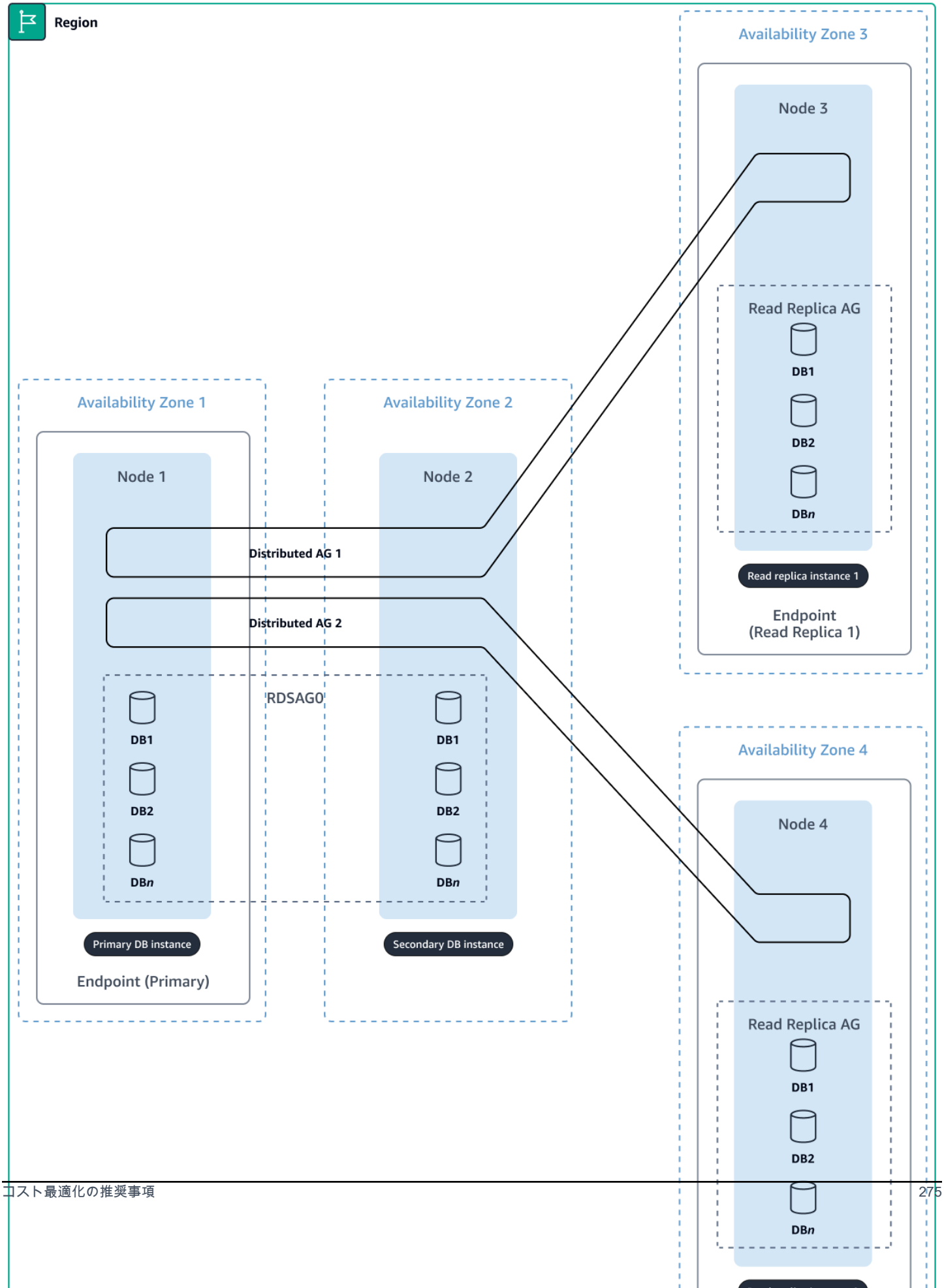
リードレプリカなしでローカルキャッシュを使用するか、Amazon RDS 上の SQL Server と並行して DAX をキャッシュレイヤーとして導入することで、大幅なコスト削減を実現できます。このレイヤーは SQL Server から負荷を軽減し、データベースの実行に必要な SQL Server のサイズを縮小します。

コスト最適化の推奨事項

ローカルキャッシュ

ローカルキャッシュは、オンプレミス環境またはクラウドでホストされているアプリケーションのコンテンツをキャッシュするために最も一般的に使用される方法の 1 つです。これは、実装が比較的簡単で直感的であるためです。ローカルキャッシュには、データベースまたは他のソースからコンテンツを取得し、より迅速にアクセスできるようにメモリまたはディスクでローカルにキャッシュすることが含まれます。このアプローチは実装は簡単ですが、一部のユースケースには適していません。例えば、これには、アプリケーションの状態やユーザーの状態を保持するなど、キャッシュコンテンツを経時的に保持する必要がある場合のユースケースが含まれます。もう 1 つのユースケースは、キャッシュされたコンテンツに他のアプリケーションインスタンスからアクセスする必要がある場合です。

次の図は、4 つのノードと 2 つのリードレプリカを持つ高い可用性の SQL Server クラスターを示しています。


Region
Availability Zone 3
Node 3
Read Replica AG
DB1
DB2
DBn
Read replica instance 1
Endpoint
(Read Replica 1)
Availability Zone 1
Availability Zone 2
Node 1
Node 2
Distributed AG 1
Distributed AG 2
RDSAG0
DB1
DB2
DBn
DB1
DB2
DBn
Primary DB instance
Secondary DB instance
Endpoint (Primary)
Availability Zone 4
Node 4
Read Replica AG
DB1
DB2
DBn

ローカルキャッシュでは、複数の EC2 インスタンス間でトラフィックの負荷分散が必要になる場合があります。各インスタンスは、独自のローカルキャッシュを維持する必要があります。キャッシュがステートフル情報を保存する場合、データベースへの定期的なコミットが必要であり、ユーザーは後続のリクエスト (スティッキーセッション) ごとに同じインスタンスに転送される必要がある場合があります。これは、アプリケーションのスケーリングを試みる際に課題となります。トラフィックの不均等な分散により、一部のインスタンスが過剰に利用される一方で、一部のインスタンスは十分に利用されない可能性があるためです。

.NET アプリケーションには、インメモリまたはローカルストレージ使用のいずれかでローカルキャッシュを使用できます。そのためには、ディスクにオブジェクトを保存して必要に応じて取得するか、データベースからデータをクエリしてメモリに保持する機能を追加できます。例えば、C# で SQL Server からのデータの、メモリおよびローカルストレージでのローカルキャッシュを実行するには、MemoryCache ライブラリと LiteDB ライブラリの組み合わせを使用できます。MemoryCache はインメモリキャッシュを提供し、LiteDB は高速で軽量な NoSQL ディスクベースの埋め込みデータベースです。

インメモリキャッシュを実行するには、.NET ライブラリ System.Runtime.MemoryCache を使用します。次のコード例は、System.Runtime.Caching.MemoryCache クラスを使用して、メモリ内にデータをキャッシュする方法を示しています。このクラスは、アプリケーションのメモリにデータを一時的に保存する方法を提供します。これにより、データベースや API など、より高価なリソースからデータを取得する必要がなくなり、アプリケーションのパフォーマンスを向上させることができます。

コードの仕組みは次のとおりです。

1. _memoryCache という名前の MemoryCache のプライベート静的インスタンスが作成されます。キャッシュには、識別するための名前 (dataCache) が付けられます。そして、キャッシュがデータを保存して取得します。
2. GetData メソッドは、string キーと、getData という Func<T> デリゲートの 2 つの引数を取るジェネリックメソッドです。キーは、キャッシュされたデータを識別するために使用され、getData デリゲートは、データがキャッシュに存在しない場合に実行されるデータ取得ロジックを表します。
3. メソッドはまず、_memoryCache.Contains(key) メソッドを使用してキャッシュにデータが存在するかどうかを確認します。データがキャッシュにある場合、メソッドは _memoryCache.Get(key) を使用してデータを取得し、期待されるタイプ T にキャストします。
4. データがキャッシュにない場合、メソッドは getData デリゲートを呼び出してデータを取得します。そして、_memoryCache.Add(key, data, DateTimeOffset.Now.AddMinutes(10))

を使用してキャッシュにデータを追加します。この呼び出しは、キャッシュエントリが 10 分後に期限切れになることを指定し、その時点でデータは自動的にキャッシュから削除されます。

5. ClearCache メソッドは string キーを引数として受け取り、_memoryCache.Remove(key) を使用して、そのキーに関連付けられたデータをキャッシュから削除します。

```
using System;
using System.Runtime.Caching;

public class InMemoryCache
{
    private static MemoryCache _memoryCache = new MemoryCache("dataCache");

    public static T GetData<T>(string key, Func<T> getData)
    {
        if (_memoryCache.Contains(key))
        {
            return (T)_memoryCache.Get(key);
        }

        T data = getData();
        _memoryCache.Add(key, data, DateTimeOffset.Now.AddMinutes(10));

        return data;
    }

    public static void ClearCache(string key)
    {
        _memoryCache.Remove(key);
    }
}
```

次のコードを使用できます。

```
public class Program
{
    public static void Main()
    {
        string cacheKey = "sample_data";

        Func<string> getSampleData = () =>
        {
```

```
            // Replace this with your data retrieval logic
            return "Sample data";
        };

        string data = InMemoryCache.GetData(cacheKey, getSampleData);
        Console.WriteLine("Data: " + data);
    }
}
```

次の例は、LiteDB を使用してローカルストレージにデータをキャッシュする方法を示しています。LiteDB は、インメモリキャッシュの代替または補完として使用できます。次のコードは、LiteDB ライブラリを使用してローカルストレージにデータをキャッシュする方法を示しています。LocalStorageCache クラスには、キャッシュを管理するための主要な関数が含まれています。

```
using System;
using LiteDB;

public class LocalStorageCache
{
    private static string _liteDbPath = @"Filename=LocalCache.db";

    public static T GetData<T>(string key, Func<T> getData)
    {
        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection<T>("cache");
            var item = collection.FindOne(Query.EQ("_id", key));

            if (item != null)
            {
                return item;
            }
        }

        T data = getData();

        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection<T>("cache");
            collection.Upsert(new BsonValue(key), data);
        }
```

```
        return data;
    }

    public static void ClearCache(string key)
    {
        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection("cache");
            collection.Delete(key);
        }
    }
}

public class Program
{
    public static void Main()
    {
        string cacheKey = "sample_data";

        Func<string> getSampleData = () =>
        {
            // Replace this with your data retrieval logic
            return "Sample data";
        };

        string data = LocalStorageCache.GetData(cacheKey, getSampleData);
        Console.WriteLine("Data: " + data);
    }
}
```

頻繁に変更されない静的キャッシュまたは静的ファイルがある場合は、これらのファイルを Amazon Simple Storage Service (Amazon S3) オブジェクトストレージに保存することもできます。アプリケーションは、起動時に静的キャッシュファイルを取得してローカルで使用できます。.NET を使用して Amazon S3 からファイルを取得する方法の詳細については、Amazon S3 ドキュメントの「オブジェクトのダウンロード」を参照してください。

DAX を使用したキャッシュ

すべてのアプリケーションインスタンスで共有できるキャッシュレイヤーを使用できます。DynamoDB Accelerator (DAX) は、DynamoDB 用のフルマネージドで可用性の高いインメモリキャッシュで、10 倍のパフォーマンス向上を実現できます。DAX を使用すると、DynamoDB テーブルで読み取りキャパシティユニットを過剰にプロビジョニングする必要性を減らすことでコストを

削減できます。これは、読み取り負荷が高く、個々のキーに対して繰り返し読み取りが必要なワークロードに特に役立ちます。

DynamoDB は、オンデマンドで、またはプロビジョニングされたキャパシティに基づいて料金が決まるため、1 か月あたりの読み取りと書き込みの回数がコストに影響します。読み取り負荷の高いワークロードがある場合、DAX クラスターは、DynamoDB テーブルの読み取り回数を減らすことでコストを削減できます。DAX を設定する方法については、DynamoDB ドキュメントの「DynamoDB Accelerator (DAX) とインメモリアクセラレーション」を参照してください。.NET アプリケーション統合の詳細については、YouTube の「Integrating Amazon DynamoDB DAX into Your ASP.NET Application」をご覧ください。

その他のリソース

- 「DynamoDB Accelerator (DAX) とインメモリアクセラレーション - Amazon DynamoDB」(DynamoDB ドキュメント)
- 「Integrating Amazon DynamoDB DAX into Your ASP.NET Application」(YouTube)
- 「オブジェクトのダウンロード」(Amazon S3 ドキュメント)

サーバーレス .NET を検討する

概要:

サーバーレスコンピューティングは、アプリケーションを構築およびデプロイするための一般的なアプローチとなっています。これは主に、最新のアーキテクチャを構築するときにサーバーレスアプローチが提供するスケーラビリティと俊敏性によるものです。ただし、一部のシナリオでは、サーバーレスコンピューティングのコストへの影響を考慮することが重要です。

Lambda は、開発者が専用サーバーを必要とせずにコードを実行できるようにするサーバーレスコンピューティングプラットフォームです。Lambda は、インフラストラクチャコストの削減を検討している .NET 開発者にとって特に魅力的なオプションです。.NET 開発者は、Lambda を使用すると、スケーラビリティが高く、潜在的に費用対効果の高いアプリケーションを開発およびデプロイできます。サーバーレスアプローチを使用することで、開発者は、アプリケーションリクエストを処理するためにサーバーをプロビジョニングする必要がなくなります。代わりに、開発者はオンデマンドで実行される関数を作成できます。これにより、サーバーレスアプローチは、仮想マシンの実行、管理、スケーリングよりもスケーラブルで管理しやすくなり、コスト効率も向上する可能性があります。結果として、使用率の低いリソースやサーバーのメンテナンスコストを心配する必要はなく、アプリケーションが使用するリソースに対してのみ料金が発生することになります。

開発者は、最新のクロスプラットフォームの .NET バージョンを使用して、高速かつ効率的で費用対効果の高いサーバーレスアプリケーションを構築できます。.NET Core 以降のバージョンは無料のオープンソースフレームワークであり、以前の .NET Framework バージョンよりもサーバーレスプラットフォームでの実行に適しています。これにより、開発者は開発時間を短縮し、アプリケーションのパフォーマンスを向上させることができます。最新の .NET では、C# や F# など、さまざまなプログラミング言語もサポートされています。このため、クラウドで最新のアーキテクチャを構築しようとしている開発者にとって魅力的なオプションです。

このセクションでは、Lambda をサーバーレスオプションとして使用してコスト削減を実現する方法について説明します。Lambda 関数の実行プロファイルの微調整、Lambda 関数のメモリ割り当ての適切なサイズ設定、ネイティブ AOT の使用、Graviton ベースの関数への移行により、コストをさらに最適化できます。

コストへの影響

コストを削減できる量は、サーバーレス関数が実行する実行数や各関数のメモリ量、期間など、いくつかの要因によって異なります。 は、1 か月あたり 100 万件の無料リクエストと 1 か月あたり 400,000 GB 秒のコンピューティング時間を含む無料利用枠 AWS Lambda を提供します。これらの無料利用枠の制限内またはその付近のワークロードの月額コストを大幅に削減できます。

Lambda 関数をターゲットとするロードバランサーを使用する場合、追加料金が発生することもあります。これは、Lambda ターゲットのロードバランサーによって処理されたデータの量として計算されます。

コスト最適化の推奨事項

Lambda 関数のサイズを適切に設定する

適切なサイズ設定は、.NET ベースの Lambda 関数のコスト最適化に不可欠なプラクティスです。このプロセスでは、コードを変更することなく、パフォーマンスとコスト効率のバランスを取る最適なメモリ設定を特定します。

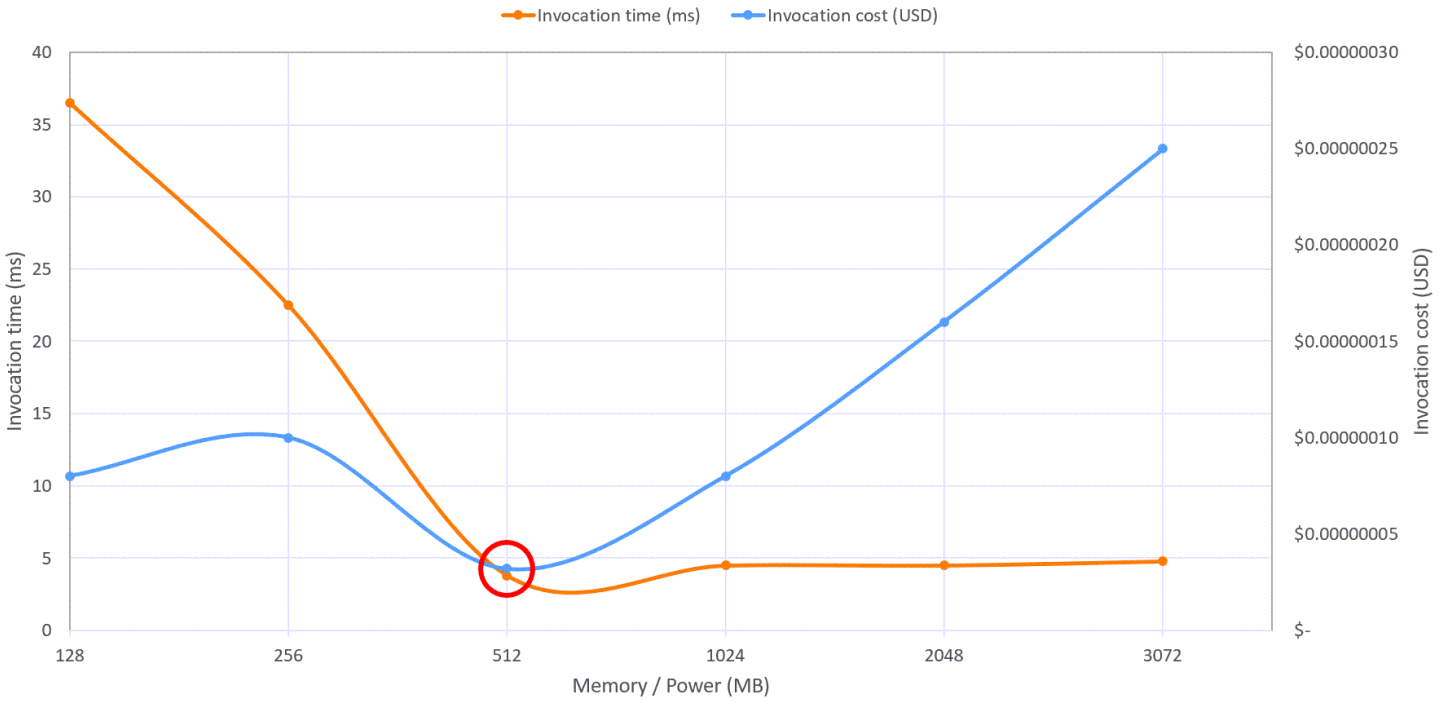
Lambda 関数のメモリを 128 ~ 10,240 MB の範囲で設定することで、呼び出し中に使用可能な vCPU の量も調整できます。これにより、メモリまたは CPU バウンドアプリケーションが実行中に追加のリソースにアクセスできるようになり、呼び出し時間と全体的なコストが削減される可能性があります。

ただし、特に変更が頻繁に行われる場合、.NET ベースの Lambda 関数に最適な設定を特定することは、手動および時間のかかるプロセスになる可能性があります。AWS Lambda Power Tuning ツー

ルは、サンプルペイロードに対して一連のメモリ設定を分析することで、適切な設定を特定するのに役立ちます。

例えば、.NET ベースの Lambda 関数のメモリを増やすと、パフォーマンスに影響を与えずに合計呼び出し時間が短縮され、コストを削減できます。関数に最適なメモリ設定は異なる場合があります。AWS Lambda Power Tuning ツールは、各関数の最も費用対効果の高い設定を特定するのに役立ちます。

次のグラフ例では、この Lambda 関数のメモリが増加すると、合計呼び出し時間が改善されます。これにより、関数の元のパフォーマンスに影響を与えることなく、合計実行のコストを削減できます。この関数の場合、関数の最適なメモリ設定は 512 MB です。これは、各呼び出しの合計コストに対してリソース使用率が最も効率的になるためです。これは関数ごとに異なり、Lambda 関数でこのツールを使用すると、適切なサイズ設定のメリットがあるかどうかを特定できます。



新しい更新がリリースされたときの統合テストの一環として、この演習を定期的に完了することをお勧めします。更新頻度が低い場合は、この演習を定期的に実行して、関数が調整され、適切なサイズに設定されていることを確認します。Lambda 関数に適したメモリ設定を特定したら、適切なサイズ設定をプロセスに追加できます。 AWS Lambda Power Tuning ツールは、新しいコードのリリース中に CI/CD ワークフローで使用できるプログラムによる出力を生成します。これにより、メモリ設定を自動化できます。

AWS Lambda Power Tuning ツールは無料でダウンロードできます。ツールの使用方法については、GitHub の「How to execute the state machine」を参照してください。

Lambda はネイティブ AOT もサポートしているため、.NET アプリケーションをプリコンパイルできます。これにより、.NET 関数の実行時間を短縮することでコストを削減できます。ネイティブ AOT 関数の作成の詳細については、Lambda ドキュメントの「.NET Lambda 関数コードをネイティブランタイム形式にコンパイルする」を参照してください。

アイドル待機時間を回避する

Lambda 関数の実行時間は、請求の計算に使用される 1 つのディメンションです。関数コードがブロッキング呼び出しを行うと、レスポンスの受信を待機する時間に対して課金されます。この待機時間は、Lambda 関数が連鎖している場合や、関数が他の関数のオーケストレーターとして動作している場合に長くなる可能性があります。バッチオペレーションや注文配送システムなどのワークフローがある場合は、管理オーバーヘッドが追加されます。さらに、Lambda の最大タイムアウトである 15 分以内にすべてのワークフローロジックとエラー処理を完了できない場合があります。

関数コードでこのロジックを処理する代わりに、ワークフローのオーケストレーターとして AWS Step Functions を使用するソリューションを再設計することをお勧めします。標準ワークフローを使用する場合は、ワークフローの合計期間ではなく、ワークフロー内の各状態遷移に対して課金されます。さらに、再試行、待機条件、エラーワークフロー、コールバックのサポートを状態条件に移行して、Lambda 関数がビジネスロジックに集中できるようにすることができます。詳細については、AWS コンピューティングブログのAWS Lambda 「コストの最適化 – パート 2」を参照してください。

Graviton ベースの関数に移行する

次世代 Graviton2 プロセッサで動作する Lambda 関数が一般利用可能になりました。ARM ベースのプロセッサアーキテクチャを使用する Graviton2 関数は、さまざまなサーバーレスワークロードのコストを 20% 削減し、パフォーマンスを最大 19% 向上させるように設計されています。レイテンシーが低く、パフォーマンスが向上する Graviton2 プロセッサで動作する関数は、ミッションクリティカルなサーバーレスアプリケーションの駆動に最適です。

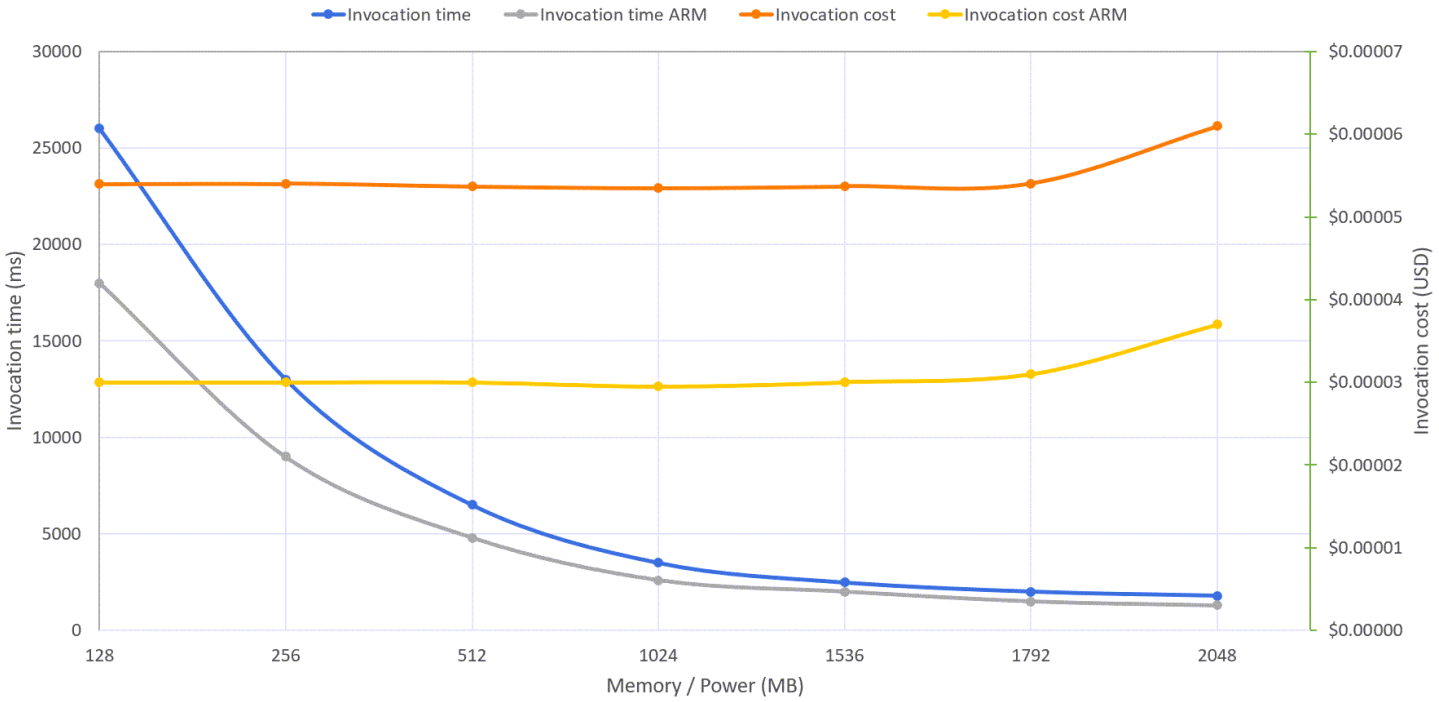
Graviton ベースの Lambda 関数への移行は、Lambda コストを最適化しようとしている .NET 開発者にとってコスト効率の高いオプションとなります。Graviton ベースの関数は、従来の x86 プロセッサの代わりに ARM ベースのプロセッサを使用します。これにより、パフォーマンスを犠牲にすることなく、大幅なコスト削減につながる可能性があります。

Graviton ベースの関数への移行にはいくつかの利点がありますが、考慮することをお勧めする課題と考慮事項もいくつかあります。例えば、Graviton ベースの関数では Amazon Linux 2 を使用する必要がありますが、これはすべての .NET アプリケーションと互換性がない場合があります。さら

に、ARM ベースのプロセッサと互換性のないサードパーティー製ライブラリや依存関係との互換性に問題がある可能性があります。

.NET Framework アプリケーションを実行し、Lambda でサーバーレスを活用する場合は、Porting Assistant for .NET を使用してアプリケーションを最新の .NET に移植することを検討できます。これにより、レガシー .NET アプリケーションの最新の .NET への移植を高速化し、アプリケーションを Linux で実行できるようになります。

次のグラフは、素数を計算する関数の x86 アーキテクチャと ARM/Graviton2 アーキテクチャの結果を比較しています。



関数は 1 つのスレッドを使用しています。メモリが 1.8 GB で設定されている場合、両方のアーキテクチャの最小期間が報告されます。さらに、Lambda 関数は複数の vCPU にアクセスできますが、この場合、関数は追加の処理能力を利用できません。同じ理由で、コストは最大 1.8 GB のメモリで安定しています。メモリが増えると、このワークロードには追加のパフォーマンス上の利点がないため、コストが増加します。Graviton2 プロセッサは、明らかにこの計算集約型関数のパフォーマンスを向上させ、コストを削減します。

ARM ベースのプロセッサを Graviton で使用するように関数を設定するには、以下を実行します。

1. にサインインし AWS マネジメントコンソール 、Lambda コンソールを開きます。
2. [関数の作成] を選択してください。

3. [関数名] に名前を入力します。
4. [ランタイム] で、[.NET 6 (C#/PowerShell)] を選択します。
5. [アーキテクチャ] で、[arm64] を選択します。
6. 必要な追加の設定を行い、[関数を作成] を選択します。

その他のリソース

- ターゲットとしての Lambda 関数 (AWS ドキュメント)
- を使用した AWS Lambda コストとパフォーマンスの最適化 AWS Compute Optimizer (AWS コンピューティングブログ)
- AWS Lambda コストの最適化 – パート 1 (AWS コンピューティングブログ)
- AWS Lambda コストの最適化 – パート 2 (AWS コンピューティングブログ)
- .NET 7 AWS Lambda を使用した でのサーバーレス .NET アプリケーションの構築 (AWS コンピューティングブログ)

目的別データベースを検討する

概要:

Microsoft ベースのワークロードを実行する上で最もコストのかかる側面の 1 つは、SQL Server などの商用データベースのライセンスに起因します。企業は、データベースプラットフォームの選択として SQL Server を標準とすることが多く、それが組織の開発文化に深く根付いています。開発者は通常、ユースケースに関係なく、リレーショナル SQL Server ベースのモデルを選択します。その理由には次のようなものがあります。

- 企業は既に SQL Server のインスタンスやライセンスが利用可能になっている。
- チームは、共有ライブラリ、ORM、ビジネスロジックを使用した SQL プログラミングモデルに慣れている。
- 経営者側が代替案を認識していない。
- 開発者が代替案を認識していない。

目的別データベースは、ユースケースのデータアクセスパターンに対応できます。これらのデータベースは、より最新のアーキテクチャ (マイクロサービスなど) が採用され、個々のアプリケーションの範囲が狭まるにつれて、企業でますます採用されています。

目的別データベースは、リレーショナルモデルを除外したり、NoSQL (非リレーショナル) モデルを必要としたりしません。実際、リレーショナルデータベースは、ワークロードの特定のニーズに応じて選択されると、目的別であると見なされます。目的別データベースを使用すると、チームは .NET アプリケーションに関連するデータベースコストを削減しながら、スケーラビリティ、耐障害性、差別化されていない困難な作業の削減などの標準的なクラウド上の利点を得ることができます。

次の表は、 が提供する専用データベースを示しています AWS。

データベース	タイプ	特性
Amazon Aurora PostgreSQL または Amazon Aurora MySQL	リレーショナル	データが固定構造を持つユースケース リレーショナルデータベースは ACID トランザクションを通じてデータ整合性を自然に維持します。
Amazon DynamoDB	キーと値のペア	ハッシュテーブルのデータ構造を使用してデータを保存する NoSQL データベース 高性能ストレージと非構造化データの取得 ユースケースには、ユーザープロファイル、セッション状態、ショッピングカートデータが含まれます。
Amazon ElastiCache	インメモリ	非構造化データをミリ秒未満のアクセス時間でメモリに保存する高性能 NoSQL データベース ユーザーセッションなどの頻繁にアクセスされるエフェメラルデータに使用され、他の低速データストアの前の

データベース	タイプ	特性
		キャッシュレイヤーとして使用されます。 ElastiCache (Redis OSS) と ElastiCache (Memcached) の両方のサポートが含まれます
Amazon MemoryDB	耐久性のあるインメモリ	耐久性のあるストレージを備えた Redis 互換の目的別データベース
Amazon Timestream	時系列	高スループットのデータインジェストを時間順に行うように設計されたデータベース ユースケースには、モノのインターネット (IoT) アプリケーションや、メトリクスまたはテレメトリデータの保存が含まれます。
Amazon DocumentDB	ドキュメント	所定の構造や、他のデータへの強制的な関係なしにデータを保存する NoSQL データベース 多くの場合、製品カタログなどの読み取り負荷の高いワークロードに使用されます。

データベース	タイプ	特性
Amazon Neptune	グラフ	データと、データ項目間の接続の表現の両方を保持する NoSQL データベース ユースケースには、不正検出、レコメンデーションエンジン、ソーシャルアプリケーションが含まれます。
Amazon Keyspaces	ワイドカラム	Apache Cassandra に基づく高性能分散データベース ユースケースには、IoT アプリケーション、イベント処理、ゲームアプリケーションが含まれます。

目的別データベースの採用の大きな推進要因は、商用ライセンスの排除にある可能性があります。ただし、DynamoDB (オンデマンドモードを含む)、Aurora、Amazon Neptune、Amazon Keyspaces などのデータベースの自動スケーリング機能を使用すると、ピーク時の使用量ではなく、平均ケースのキャパシティをプロビジョニングできます。Timestream などの目的別データベースはサーバーレスであり、事前プロビジョニングなしで需要に合わせて自動的にスケーリングされます。

AWS は、専用のオープンソース互換リレーショナルデータベースを使用する場合、Babelfish for Aurora PostgreSQL を提供しますが、アプリケーションに大幅なコード変更を行うことができない、または行おうとしない場合があります。場合によっては、Babelfish を使用して、ほとんど変更を加えずに既存の SQL Server アクセスコードを使用できることがあります。

アプリケーションの目的別リレーショナルデータベースを選択するときは、アプリケーションに必要な機能と同じ (または機能的に同等の) 機能を保持することが重要です。この推奨事項では、アプリケーションのプライマリデータストアとして目的別データベースを取り上げています。特定のアプリケーション (キャッシュなど) については、他の推奨事項で言及しています。

コストへの影響

.NET ワークロードの目的別データベースを採用すると、コンピューティングの消費/コストに直接影響する可能性は低いものの、.NET アプリケーションが消費するデータベースサービスのコストに直接影響する可能性があります。実際、俊敏性、スケーラビリティ、耐障害性、データ耐久性という付加的な利点と比較して、コスト削減は副次的な目標になる可能性があります。

アプリケーションの目的別データベースを選択し、効果的に使用するためのデータ戦略を再設計する完全なプロセスを説明することは、このガイドの範囲外です。詳細については、 AWS チュートリアルディレクトリの「Purpose-built databases」を参照してください。

次の表に、SQL Server を目的別データベースに置き換えることでアプリケーションコストがどのように変化するかを示すいくつかの例を挙げます。これらは単純に概算見積もりであることに注意してください。正確な本番コストを計算するには、実際のワークロードのベンチマークと最適化が必要です。

これらは、オンデマンドコンピューティングと 100 GB SSD、`us-east-1` の単一インスタンスデータベースを含む、一般的に使用される目的別データベースの見積もりです。ライセンスコストには、SQL Server ライセンスとソフトウェア保証が含まれます。

次の表は、商用データベースの例の推定コストを示しています。

データベースエンジン	ライセンスモデル	インスタンスタイプ/スペック	AWS コンピューティング + ストレージコスト	ライセンスコスト	合計月額コスト
Amazon EC2 の SQL Server Standard Edition	ライセンスインクルード	r6i.2xlarge (8 CPU/64 GB RAM)	1,345.36 USD	0.00 USD	1,345.36 USD
Amazon EC2 の SQL Server Enterprise Edition	ライセンスインクルード	r6i.2xlarge (8 CPU/64 GB RAM)	2,834.56 USD	0.00 USD	2,834.56 USD

データベースエンジン	ライセンスモデル	インスタンスタイプ/スペック	AWS コンピューティング + ストレージコスト	ライセンスコスト	合計月額コスト
Amazon EC2 の SQL Server Standard Edition	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	644.56 USD	456.00 USD	1,100.56 USD
Amazon EC2 の SQL Server Enterprise Edition	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	644.56 USD	1,750.00 USD	2,394.56 USD
Amazon RDS の SQL Server Standard Edition		db.r6i.2xlarge (8 CPU/64 GB RAM)	2,318.30 USD	0.00 USD	2,318.30 USD
Amazon RDS の SQL Server Enterprise Edition		db.r6i.2xlarge (8 CPU/64 GB RAM)	3,750.56 USD	0.00 USD	3,750.56 USD

次の表は、目的別の例の推定コストを示しています。

データベースエンジン	インスタンスタイプ/スペック	AWS コンピューティング + ストレージコスト	ライセンスコスト	合計月額コスト
Amazon Aurora PostgreSQL	r6g.2xlarge (8 CPU/64 GB RAM)	855.87 USD	0.00 USD	855.87 USD
DynamoDB	プロビジョニング済みベース 100 WCU/400 RCU	72.00 USD		72.00 USD
Amazon DocumentDB	db.r6i.2xlarge (8 CPU/64 GB RAM)	778.60 USD		778.60 USD

> **Important**
>
> この表は、購入後の最初の 3 年間の、ソフトウェアアシュアランス付き SQL Server の推定ライセンスコストに基づいています。SQL Server Standard Edition の場合: 4,100 USD、2 コアパック、3 年間。SQL Server Enterprise Edition の場合: 15,700 USD、2 コアパック、3 年間。

目的別データベースを採用する前に、コストへの影響を考慮することをお勧めします。例えば、目的別データベースを使用するようにアプリケーションを更新するコストは、アプリケーションとソースデータベースの複雑さに関連します。このアーキテクチャの切り替えを計画するときは、総保有コストを必ず考慮してください。これには、アプリケーションのリファクタリング、新しいテクノロジーに関するスタッフのスキル向上、各ワークロードで予想されるパフォーマンスと消費の慎重な計画が含まれます。そこから、投資がコスト削減に値するかどうかを決定できます。ほとんどの場合、サポートが終了した製品を維持することはセキュリティとコンプライアンスのリスクであり、それを修復するコストは、労力と初期投資に見合う価値があります。

コスト最適化の推奨事項

SQL Server にアクセスする .NET アプリケーションには、目的別リレーショナルデータベース用の代替ライブラリがあります。これらのライブラリをアプリケーションに実装して、同様の SQL Server アプリケーションの機能を置き換えることができます。

次の表は、多くの一般的なシナリオで使用できるいくつかのライブラリを示しています。

Library	データベース	置き換え元	フレームワークの互換性
Npgsql Entity Framework Core プロバイダー	Amazon Aurora PostgreSQL	Entity Framework Core SQL Server プロバイダー	最新の .NET
Npgsql Entity Framework 6 プロバイダー	Amazon Aurora PostgreSQL	Entity Framework 6.0 SQL Server プロバイダー	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。
Npgsql (ADO.NET 互換 PostgreSQL ライブラリ)	Amazon Aurora PostgreSQL	ADO.NET	.NET Framework/最新の .NET
MySQL Entity Framework Core プロバイダー	Amazon Aurora MySQL	Entity Framework Core SQL Server プロバイダー	最新の .NET
Pomelo.EntityFrameworkCore.MySql	Amazon Aurora MySQL	Entity Framework Core SQL Server プロバイダー	最新の .NET

Babelfish を使用して Amazon Aurora PostgreSQL に接続する場合、接続に特別なコーディングは必要ありません。ただし、使用前にすべてのコードを徹底的にテストする必要があります。

その他の目的別データベースには、目的別データベースにアクセスできるようにする、.NET 互換ライブラリにアクセスするためのライブラリがあります。以下に例を示します。

- Amazon DynamoDB NoSQL データベースの使用 (AWS SDK for .NET ドキュメント)
- 「MongoDB C# ドライバー」(MongoDB ドキュメント)
- 「.NET」(Timestream ドキュメント)
- 「Using a Cassandra .NET Core client driver to access Amazon Keyspaces programmatically」(Amazon Keyspaces ドキュメント)
- 「Using .NET to connect to a Neptune DB instance」(Neptune ドキュメント)

目的別に構築されたデータベースに移行する場合は、 からこれらのツールを使用して移行プロセス AWS に役立てることができます。

- AWS Schema Conversion Tool (AWS SCT) は、SQL Server スキーマを Amazon Aurora および Amazon DynamoDB に変換するのに役立ちます。
- AWS Database Migration Service (AWS DMS) は、SQL Server から Aurora または DynamoDB にデータを 1 回または継続的に移行するのに役立ちます。
- Babelfish Compass は、Babelfish for Aurora PostgreSQL で使用するための SQL Server データベースの互換性を確認するのに役立ちます。

その他のリソース

- 「Guidance for migrating SQL Server to Amazon Aurora PostgreSQL」(AWS データベースブログ)
- Babelfish APP Modernization Immersion Day (AWS Workshop Studio)
- .NET Immersion Day (AWS Workshop Studio)
- 「Getting started with Amazon Timestream with .NET」(GitHub)
- での最新の .NET アプリケーション専用のデータベース AWS (AWS プレゼンテーション)

次のステップ

このガイドを確認したら、MACO を実装するために以下の次のステップを実行することをお勧めします。

1. MACO エキスパートに問い合わせます。MACO エキスパートは、疑問を解消し、懸念事項に対処するお手伝いをします。既に AWS アカウントチームと連携している場合は、チームに連絡して MACO エキスパートのサポートを依頼してください。アカウントチームがない場合は、optimize-microsoft@amazon.com にご連絡ください。
2. 推奨事項を適用します。このガイドや MACO エキスパートとの話し合いで学んだ推奨事項、ベストプラクティス、戦略を適用します。
3. コストの変化を追跡します。ワークロードにタグを付け、AWS Cost Explorer や AWS Budgets などのサービスを使用して、詳細なコスト追跡、モニタリング、制御を行います。

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィードをサブスクライブできます。

変更	説明	日付
SQL Server の更新	SQL Server ワークロードの CPU の最適化セクションを更新し、Amazon EC2 インスタンスの CPU の最適化機能に関する詳細を追加しました。	2025 年 10 月 22 日
SQL Server の更新	「Optimize CPU for SQL Server workloads」セクションを更新しました。	2024 年 10 月 25 日
SQL Server とコンテナの更新	Compute Optimizer、SQL Server ワークロードのレコメンデーションの確認、App2Container を使用した Windows アプリケーションのリプラットフォームを使用して、SQL Server のサイズを最適化するセクションを追加しました。 Trusted Advisor App2Container	2024 年 6 月 29 日
SQL Server ライセンスの最適化	「Compute Optimizer を使用して SQL Server ライセンスを最適化する」セクションを追加しました。	2024 年 5 月 22 日
初版発行	—	2023 年 12 月 21 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範ガイダンスによって提供される戦略、ガイド、パターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

数字

7 Rs

アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。

- リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例： Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。

- 廃止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

A2A (Agent-to-Agent)

タスクの委任と状態転送をサポートするagent-to-agentコラボレーション用のステートフルプロトコル。

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。アクティブ/パッシブ移行よりも柔軟な方法ですが、さらに多くの作業が必要となります。

アクティブ/パッシブ移行

ソースデータベースとターゲットデータベースを同期させながら、データがターゲットデータベースにレプリケートされている間、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。移行中、ターゲットデータベースはトランザクションを受け付けません。

[エージェント]

目標を達成するためのツールを使用して、自律的に推論、計画、アクションを実行できる AI システム。

エージェントオペレーション

AI エージェントを本番環境で大規模に構築、テスト、デプロイ、実行するための運用プラクティス。

集計関数

複数行に処理を行い、グループ全体を対象に単一の戻り値を計算する SQL 関数。集計関数の例としては、SUM や MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーション制御

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の重要な要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」をご覧ください。

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン (AZ)

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じ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する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスを整理しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションのガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントの「動作グラフのデータ」を参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

それぞれが独立しているが、同一の環境を 2 つ作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (ブルー) で実行し、新しいアプリケーションバージョンを別の環境 (グリーン) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染しており、ボットハーダーまたはボットオペレーターと呼ばれる単一の当事者によって制御されているボットのネットワーク。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発したり、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 AWS Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータを参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「ビジネス機能を中心に組織化」セクションを参照してください。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

エンドユーザーへのバージョンリリースを、時間をかけて段階的に行うこと。確信が持てたら新規バージョンをデプロイして、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストすること。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

シチズンデベロッパー

専門的な技術スキルを持たないノーコード/ローコードプラットフォームを使用して AI アプリケーションを作成するビジネスユーザー。

クライアント側の暗号化

ターゲットがデータ AWS のサービス を受信する前のローカルでのデータの暗号化。

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的に、エッジコンピューティングに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」を参照してください。

導入のクラウドステージ

組織が、 AWS クラウドへの移行時に通常実行する 4 つの段階。

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略との関連性については、 AWS「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、GitHub や Bitbucket Cloud があります。コードの各バージョンはブランチと呼ばれます。マイクロサービスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオといった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

ワークロードにおいて、設定が想定した状態から変化すること。これによって、ワークロードが非準拠になる可能性がありますが、この状態は、徐々に生じ、意図的なものではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイ

することも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、「データ分類」を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行えるアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼できる ID のみが、期待されるネットワークから信頼できるリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには、一般的に、大量の履歴データが含まれており、多くの場合、それらはクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせます。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

深層学習

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS Organizations ドキュメントの「AWS Organizationsで使用できるサービス」を参照してください。

トラブルシューティング

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、「 AWSでのセキュリティコントロールの実装」の「検出的コントロール」を参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、クエリの制約、フィルタリング、結果セットのラベル付けに使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

ディザスタによるダウンタイムとデータ損失を最小限に抑えるための戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用す

る方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を改善できます。

電子データ交換 (EDI)

組織間で行う、ビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティング処理。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (会計、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service (AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。

- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイドを参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

スタースキーマの中央にあるテーブル。ビジネスオペレーションに関する定量的データが保存されます。一般的に、ファクトテーブルは、2 種類の列で構成されます。1 つは測定値が含まれる列、もう 1 つはディメンションテーブルへの外部キーが含まれる列です。

フェイルファスト

開発ライフサイクルを短縮するために、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

では AWS クラウド、アベイラビリティーゾーン、コントロールプレーン AWS リージョン、データプレーンなどの境界で、障害の影響を制限し、ワークロードの耐障害性を向上させるのに役立ちます。詳細については、「AWS 障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

数ショットプロンプト

LLM に、タスクと望ましい出力を示す例を少数提示した後に、類似のタスクを実行させること。この手法は、プロンプトに記述された例 (ショット) からモデルが学習する「インコンテキスト学習」の一種です。数ショットプロンプトは、特定のフォーマット、推論、専門知識が必要なタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセス制御」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

大規模な深層学習ニューラルネットワークであり、一般化およびラベル付けされていないデータからなる大規模データセットでトレーニングされています。FM により、言語理解、テキストおよび画像生成、自然言語での会話といった、一般的な各種タスクを実行できます。詳細については、「基盤モデルとは何ですか?」を参照してください。

FM ゲートウェイ

基盤モデルへのアクセスを制御および正規化する一元化された仲介者。LLM ゲートウェイとも呼ばれます。

G

生成 AI

AI モデルのサブセット。大量のデータでトレーニングされており、シンプルなテキストプロンプトを使用して、画像、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できます。詳細については、「生成 AI とは何ですか?」を参照してください。

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リストを使って指定します。詳細については、CloudFront ドキュメントの「コンテンツの地理的ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、Amazon GuardDuty AWS Security Hub CSPM、、 AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

ガードレール (AI)

エージェントの入力と出力をフィルタリング、検証、制約する安全メカニズムは、責任ある安全な AI の動作を確保するのに役立ちます。

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

高可用性 (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニング用データセットから保留される、ラベル付き履歴データの一部。ホールドアウトデータを使用すると、モデル予測をホールドアウトデータと比較して、モデルのパフォーマンスを評価できます。

ヒューman-in-the-loop (HitL)

エージェントの実行が重要な決定時点で人間によるレビューと承認のために一時停止するワークフローパターン。

同種データベースの移行

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

アイドル状態のアプリケーション

90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、ミュータブルインフラストラクチャよりも一貫性、信頼性、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

2016 年に Klaus Schwab 氏が提唱した用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩による、ビジネスプロセスのモダナイズを意味します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

インダストリアル IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイドを参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。

大規模言語モデル (LLM)

大量のデータで事前トレーニングされた深層学習 AI モデル。LLM では、質問への回答、ドキュメントの要約、他言語へのテキスト翻訳、文を完成させるなど、さまざまなタスクを実行できます。詳細については、「大規模言語モデル (LLM) とは何ですか?」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの「最小特権アクセス許可を適用する」を参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティやプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスを招く可能性があります。マルウェアの例には、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス がインフラストラクチャレイヤー、オペレーティングシステム、プラットフォームを AWS 運用し、ユーザーがエンドポイントにアクセスしてデータを保存および取

得します。マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

「Migration Acceleration Program」を参照してください。

MCP

「モデルコンテキストプロトコル」を参照してください。

モデルコンテキストプロトコル (MCP)

エージェントとツール間の通信のためのステートレスプロトコル。

MCP サーバー

Model Context Protocol を通じて 1 つ以上のツールを公開するサービス。

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。詳細については、AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

発行/サブスクライブのパターンに基づく、軽量のマシンツーマシン (M2M) 通信プロトコルであり、リソースに限りのある IoT デバイスに使用されます。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれ

る場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS 「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイドを参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを AWS クラウドに移行するために使用するアプローチ。詳細については、この用語集の 7 Rs エントリと、「組織を動員して大規模な移行を加速する」を参照してください。

ML

「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「AWS クラウドでのアプリケーションのモダナイズ戦略」を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定された

ギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、「モノリスをマイクロサービスに分解する」を参照してください。

MPA

「Migration Portfolio Assessment」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルなインフラストラクチャ

本番ワークロードに使用する既存のインフラストラクチャを更新および変更するためのモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

Open Process Communications - Unified Architecture (OPC-UA)

産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。OPC-UA により、相互運用の際に、データ暗号化、認証、認可の各スキームを標準化できます。

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイドを参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail をログに記録する によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの「組織の証跡の作成」を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイドを参照してください。

オリジンアクセス制御 (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OAC も併せて参照してください。OAC では、より詳細な、強化されたアクセス制御が可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されたネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

「個人を特定できる情報」を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

次の操作を可能にするオブジェクト: アクセス許可を定義する (ID ベースのポリシーを参照)。アクセス条件を指定する (リソースベースのポリシーを参照)。 AWS Organizations の組織における全アカウントにアクセス許可の上限を定義する (サービスコントロールポリシーを参照)。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行の準備状況の評価」を参照してください。

述語

`true` または `false` を返すためのクエリ条件。一般的に、`WHERE` 句に記述されます。

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、「AWSでのセキュリティコントロールの実装」の「予防的コントロール」を参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントの「ロールに関する用語と概念」にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

プライベートホストゾーン

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠リソースのデプロイ防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

製品の設計、開発、発売から、成長、成熟、衰退、廃棄に至る、製品のライフサイクル全体を通してデータとプロセスを管理すること。

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

製造分野で使用される、信頼性と適応性に優れたコンピュータであり、これによって、マシンをモニタリングするとともに、製造プロセスを自動化します。

プロンプトチェイニング

1 つの LLM プロンプトによる出力を次のプロンプトの入力に使用して、より良いレスポンスを生成します。この手法を使用すると、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改良または拡張したりできます。これによって、モデルのレスポンスの精度と関連性が向上し、粒度の高いパーソナライズされた結果を得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

発行/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、マイクロサービスは、他のマイクロサービスがサブスクライブ可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

手順などの一連のステップであり、SQL リレーショナルデータベースシステムのデータにアクセスするために使用されます。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RAG

「検索拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の とは独立しています。詳細については、「アカウントが使用できる AWS リージョン を指定する」を参照してください。

リグレッション

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートが含まれる場合は RASCI マトリックスと呼ばれ、含まれない場合は RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」を参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、トレーニングデータソースの外部にある信頼できるデータソースが参照されます。例えば、RAG モデルによって、組織のナレッジベースまたはカスタムデータのセマンティック検索を実行できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

定期的にシークレット情報を更新して、攻撃者が認証情報にアクセスするのをより困難にするプロセス。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 にログイン AWS マネジメントコンソール したり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントの「SAML 2.0 ベースのフェデレーションについて」を参照してください。

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存する AWS Secrets Managerパスワードやユーザー認証情報などの機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値には、バイナリ、1 つの文字列、複数の文字列を指定できます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの概要」を参照してください。

セキュリティバイデザイン

開発プロセス全体を通してセキュリティが考慮されているシステムエンジニアリングのアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントへの自動レスポンスまたは自動修復を目的として、事前定義およびプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例には、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先で、それ AWS のサービス を受け取る によるデータの暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、「AWS 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

サービスレベルインジケータによって測定され、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、「責任共有モデル」を参照してください。

シャドウ AI

組織内の管理対象チャネルの外部で構築または使用される認可されていない AI アプリケーション。

SIEM

「Security Information and Event Management システム」を参照してください。

単一障害点 (SPOF)

特定のアプリケーションを構成する単一の重要なコンポーネントで発生し、システム稼働に支障をきたす可能性のある障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お

客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

データベースの編成構造を意味し、1 つの大きいファクトテーブルにトランザクションデータまたは測定データが保存され、1 つ以上の小さいディメンションテーブルにデータ属性が保存されます。この構造は、データウェアハウスやビジネスインテリジェンスを用途とするように設計されています。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視制御とデータ取得 (SCADA)

製造分野において、ハードウェアとソフトウェアを使用して物理アセットと本番運用をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に動作を指示する手法。システムプロンプトは、コンテキストを設定して、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

tool

エージェントが外部システムでオペレーションを実行するために呼び出すことができる関数または API。

トランジットゲートウェイ

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザを分け合えることができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。

未分化なタスク

ヘビーリフティングとも呼ばれ、アプリケーションの作成と運用には必要だが、エンドユーザーに直接的な価値をもたらさなかったり、競争上の優位性をもたらしたりしない作業です。未分化なタスクの例としては、調達、メンテナンス、キャパシティプランニングなどがあります。

上位環境

「環境」を参照してください。

V

バキューミング

ストレージを再利用してパフォーマンスを向上させるために、増分更新後にクリーンアップを行うデータベースのメンテナンス操作。

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

プライベート IP アドレスを使用してトラフィックをルーティングできる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに何らかの形で関連している行のグループに計算を実行する SQL 関数。ウィンドウ関数は、移動平均を計算したり、現在の行の相対位置に基づいて他の行の値にアクセスするといったタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

データを 1 回のみ書き込むことで、データの削除や変更を防ぐストレージモデル。承認済みユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャは、イミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

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。