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の最適化 AWS

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

2024 年 10 月 (ドキュメント履歴)

概要

このガイドでは、 上の 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、ビジネス目的を満たす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礎 AWS 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。このガイダンスを総称して Microsoft on AWS Cost Optimization (MACO) と呼びます。MACO ガイダンスは業界の専門家によって開発され、実際のシナリオに基づいています。

このガイドでは、次の Microsoft ワークロードについて説明します。

- Amazon Elastic Compute Cloud の Windows (Amazon EC2)
- SQL サーバー
- コンテナ
- ストレージ
- アクティブディレクトリ
- .NET

対象者

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。 AWS 請求、Microsoft テクノロジー、および AWS システム管理に関する経験と基本的な理解があることは有益ですが、必須ではありません。

このガイドの使い方

このガイドを使用して、クラウドMACOへの移行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み、 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 の Windows EC2
- SQL サーバー
- コンテナ
- [Storage (ストレージ)]
- Active Directory
- .NET

> **Important**
>
> このガイドで提供されているコードサンプルは、デモンストレーションのみを目的としています。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のがベストプラクティスです。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、 を使用してコードから生じるコストの変化を確認することをお勧めしますAWS Cost Explorer。これにより、エッジケースやその他の問題をトラブルシューティングし、後で問題が発生する可能性があります。

> **Important**
>
> このガイドの料金例は、発行時の料金に基づいています。料金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、お客様とお客様の組織が以下のビジネス成果を達成するのに役立ちます。

- AWS 最適化とライセンス評価 (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、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について説明します。
- Microsoft ワークロードの 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) 上の Windows、SQL Server、コンテナ、ストレージ、Active Directory、.NET のワークロードなど、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、 を使用したクラウド財務管理に重点を置き、MACO の推奨事項とベストプラクティスに準拠するクラウド AWS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。

以下の大まかなタスクは、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、「 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
- 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、請求情報とコスト管理ユーザーガイドの AWS Budgets「 によるコストの管理」を参照してください。AWS
- 最適化 AWS とライセンス評価 (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 「最適化とライセンスの評価」を参照してください。
- を使用して Windows および SQL Server ワークロード用の Bring Your Own License (BYOL) を使用します Amazon Elastic Compute Cloud Dedicated Hosts。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の持ち込み」セクションを参照してください。
- で SQL Server ライセンスを最適化します AWS。詳細については、このガイドの「SQL サーバー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 サーバー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。

- で EC2 インスタンススケジューラを使用してワークロードを制御します AWS。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、本番稼働以外のワークロードの 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。
- Amazon FSx for Windows File Server の開発およびテストワークロードには、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「1 つ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。
- を使用して Windows ワークロードを適正化します AWS Compute Optimizer。詳細については、このガイドの「適切なサイズの Windows ワークロード」セクションを参照してください。
- Savings Plans を使用して、Amazon EC2 の Windows の支出を最適化します。 Savings Plans 詳細については、このガイドの「Amazon での Windows の支出を最適化する EC2」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 でデータ重複排除を有効にする FSx」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 サーバーバックアップ戦略の最適化」セクションを参照してください。
- 静的な .NET Framework アプリケーション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、このガイド静的 の動的スケーリングをサポートします。NET フレームワークアプリケーションの「」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 を検討してください。NET」セクションを参照してください。
- Windows アプリケーション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。
- を使用してAWS Compute Optimizer、 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正化します。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 にリファクタリングし、NETLinux 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 データベースを最新化します。詳細については、このガイドの「SQL サーバー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「アクティブディレクトリ」セクションを参照してください。

による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、 AWS ホワイトペーパー「 クラウド財務管理機能」を参照してください。

コストを最適化するための上位のレコメンデーション

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。このガイドではコスト最適化に関する推奨事項を示しますが、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。これらのレコメンデーションは迅速に実装でき、組織に大きな影響を与えます。これらのレコメンデーションは、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。

上位のレコメンデーション

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。「実装が難しい」列は、実装が最も簡単な (1) から実装が最も難しい (5) までのスケールに基づいて、各最適化を評価します。「削減額の見積もり」列には、推奨される最適化ごとに組織が削減できる削減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
適切なサイズの Windows ワークロード	3	25%
Windows および SQL Server ワークロードのライセンスの持ち込み	3	30%
SQL Server Developer Edition を評価する	2	20%
SQL サーバー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 にリファクタリングし、NETLinux に移行する	5	10～20%
Amazon での Windows の支出を最適化する EC2	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**
>
> 上の表の推定削減額は、アカウント内の全体的な AWS 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、インスタンススケジューラをさまざまな環境タイプとサイズに実装して、潜在的な削減額を変更することができます。見積りは、特に Amazon EC2 インスタンスのコストに適用され、他の の全体的な節約を意味するものではありません AWS のサービス。これらの見積もりは、保証ではなくゲージとして提供されます。

MACO の専門家は、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会議を設定するには、 アカウントチームに問い合わせるか、optimize-microsoft@amazon.com まで E メールでお問い合わせください。

AWS 最適化とライセンス評価

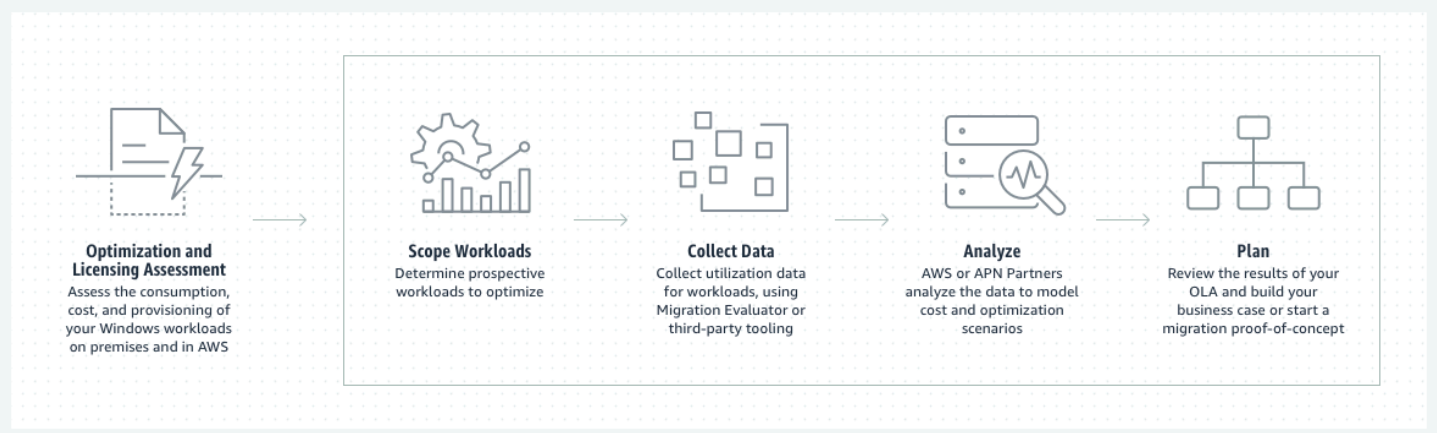
概要

AWS Optimization and Licensing Assessment (AWS OLA) は、リソース使用率、サードパーティーのライセンス、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および既存のクラウド環境を評価および最適化するのに役立ちます。 AWS OLA を使用すると、 で既存の Microsoft ワークロードに移行 AWS または評価する際にコスト削減を可能にする移行およびライセンス戦略を組織で構築できます AWS。 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、および契約を理解します。
- リソースの適切なサイズ。
- へのロードマップを作成します AWS クラウド。
- 既存の投資を使用し、使用した分だけ支払うことで、コストを削減または排除できます。

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。を使用して AWS OLA AWS Partner Network 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レコメンデーション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

の Microsoft ワークロードには、次の 2 つの AWS OLA オプションから選択できます AWS。

- 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 にあります。には AWS 、「RVTools。その後、 は 1 ~ 5 日のターンアラウンドタイムを提供 AWS できます。このアプローチでは、VMware vCenter から直接取得された point-in-time 情報を使用してサイジングレコメンデーションを作成し、オンデマンド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 日から 30 日の使用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ツールを使用して、 と連携する analysis. AWS works を完了 AWS Partner Network し、料金モデルや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、最終的な総所有コスト (TCO) 評価を提供できます。

完全な評価

完全な AWS OLA 評価は、1 時間の電話から開始されます。このコール中に、 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設定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
- 次のステップを計画する

ワークロードの範囲

まず、自分とチーム AWS 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 (非本番環境や本番環境など) ごとに分類されます。スコープには、ワークロードの場所が含まれます。これは、 に移行するワークロード AWS、 ですでに実行されているワークロード AWS (Amazon EC2 用の AWS OLA など) 、または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

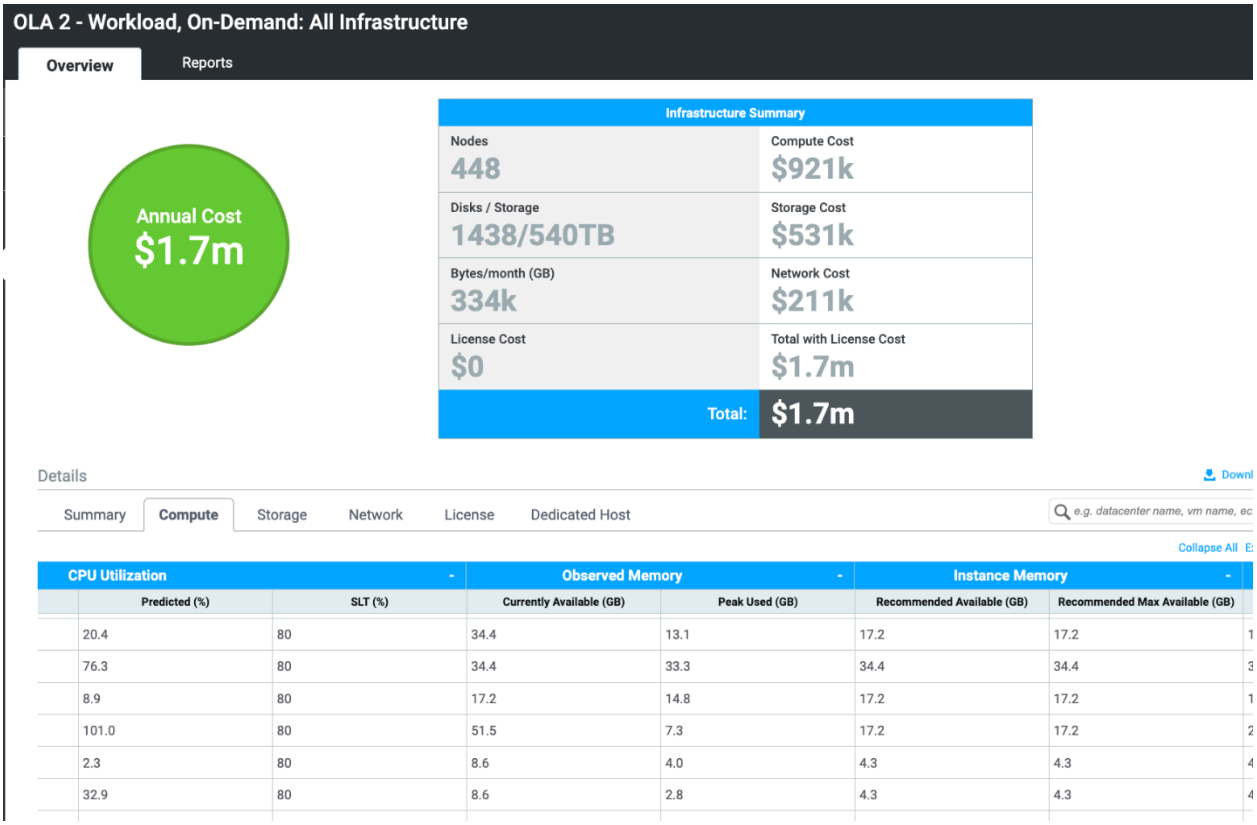
次に、 は、リソースの検出とサーバーからのパフォーマンスデータの収集に役立つツールを AWS デプロイします。このツールには、次の 4 つのデプロイオプションがあります。

- 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の収集と分析

ツールのデプロイでは、各オプションを組み合わせて一致させ、結果を統合できます。どのオプションを選択しても IT リソースに負荷がかからないようにすることが重要です。評価プロセスをできるだけターンキーにするために、 AWS は を稼働させます。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、総所有コスト (TCO) 分析とレビューのためのレコメンデーションを準備します。

データ収集には通常、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が分析されるまでに 2～3 週間かかります。理想的には、このコレクションは当月のピーク時 (例えば、 end-of-month 財務レポート中など) AWS に行われます。 は、適切なサイズの AWS インスタンスが何であるかに関する適切な統計サンプルを提供しながら、パフォーマンスがオンプレミスで利用可能なものを超えることを保証するため、ピーク時の使用状況を把握したいと考えています。 は、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使用して使用率メトリクスを AWS マージし、特定のワークロードが必要とする CPU と RAM を正確に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ターゲットよりも小さくなります。これにより、インスタンスサイズの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。

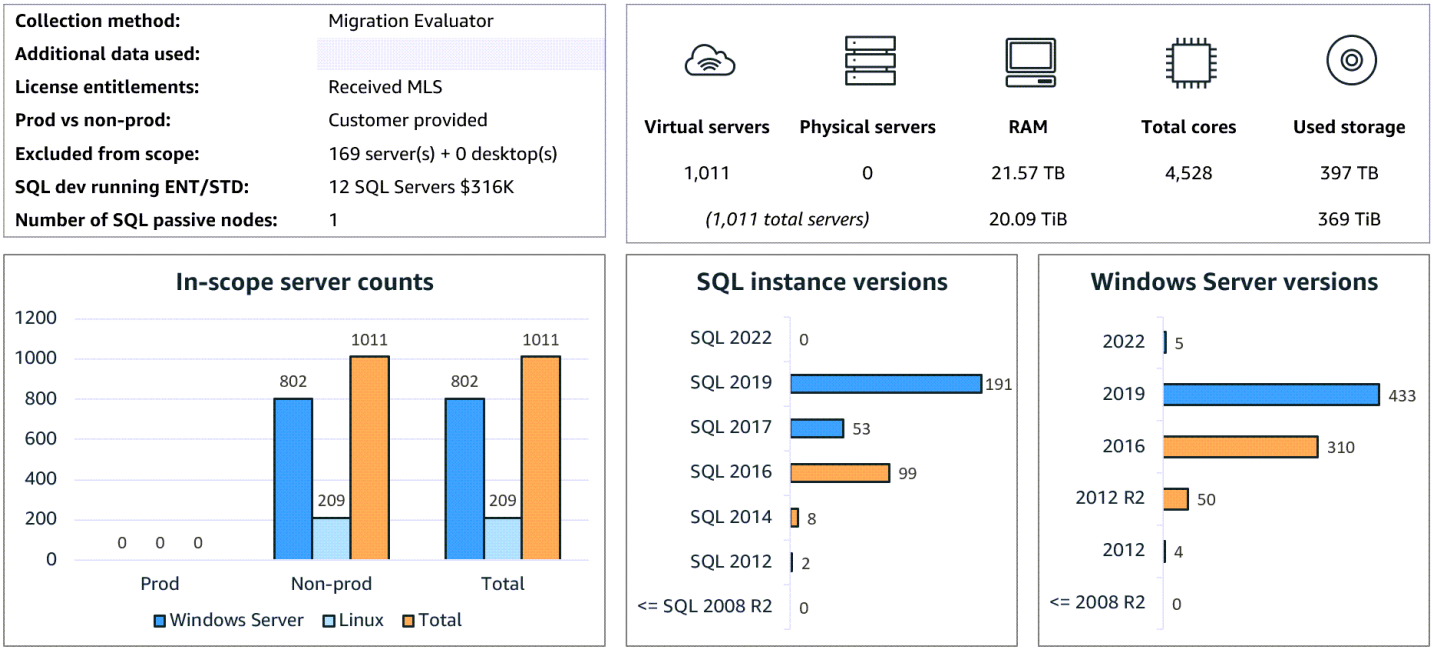
次のダッシュボードビューは、評価でキャプチャできるインフラストラクチャコストの例を示しています。



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。 はデータ AWS を確認し、検出結果を要約してから、オンプレミスの使用とクラウド移行に関するレコメンデーションを作成します。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的に調整できる) 、適切な SKU の機会 (SQL Server Enterprise Edition が使用されているが、リソース要件と機能の使用法から 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コストとライセンスコストを削減できます。コアからライセンスされている SQL Server などの製品では、多くの場合、ワークロードをより高価なコンピューティングインスタンスに配置するのが財務上の意味があります。つまり、CPU プロファイルと RAM と vCPU の比率が、ライセンスインクルードと Bring Your Own License (BYOL) の両方のユースケースのライセンスコア数を減らすために正味の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のライセンス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：

- ピーク時の使用に対してオーバープロビジョニングを避けます。
- リソースを過剰に指定したり、十分に使用したりしないようにします。
- 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します。
- マネージドデータベースに移行することで、運用コストを節約できます。

サードパーティーのライセンス削減の例：

- 同じワークロードを実行するために必要なコアを減らします。
- 不要な SQL Server Enterprise エディションとアドオンパックを削除します。
- ゾンビサーバーを削除し、古いハードウェアを置き換えます。
- BYOL とライセンス込みオプションを使用して、将来の商用契約を削減します。
- を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します。

次のステップを計画する

最後に、収集したパフォーマンスデータ AWS 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、スコープ設定された環境を集計して調べ、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、 への移行かを判断するのに役立ちます AWS。 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、クラウド経済のビジネスケースを構築できます。

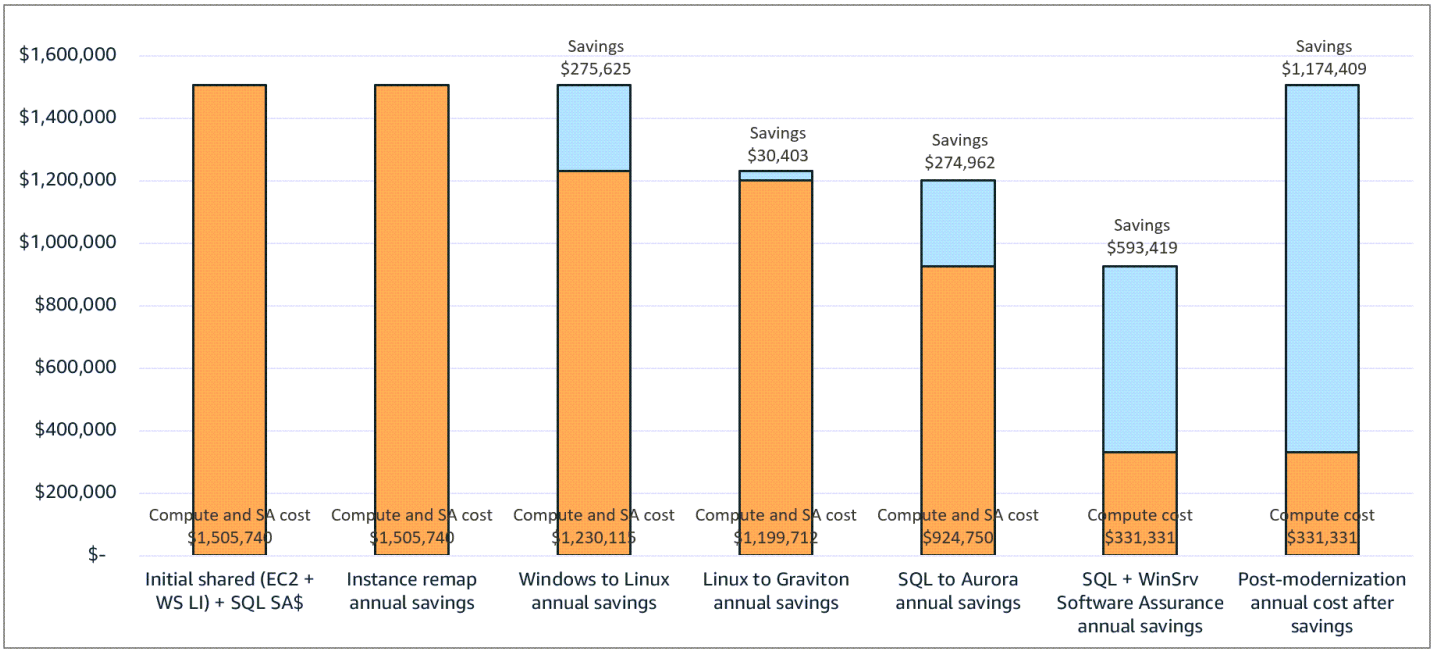
	Option 1: Amazon EC2 shared	**Option 1a: Amazon EC2 shared + power management**	**Option 2: Amazon EC2 mixed**	**Option 2a: Amazon EC2 mixed + power management**
Option details: compute	*100% Reserved Instances (RIs)*	*RIs + on-demand power management*	*100% Ris*	*RIs + on-demand power management*
Option 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

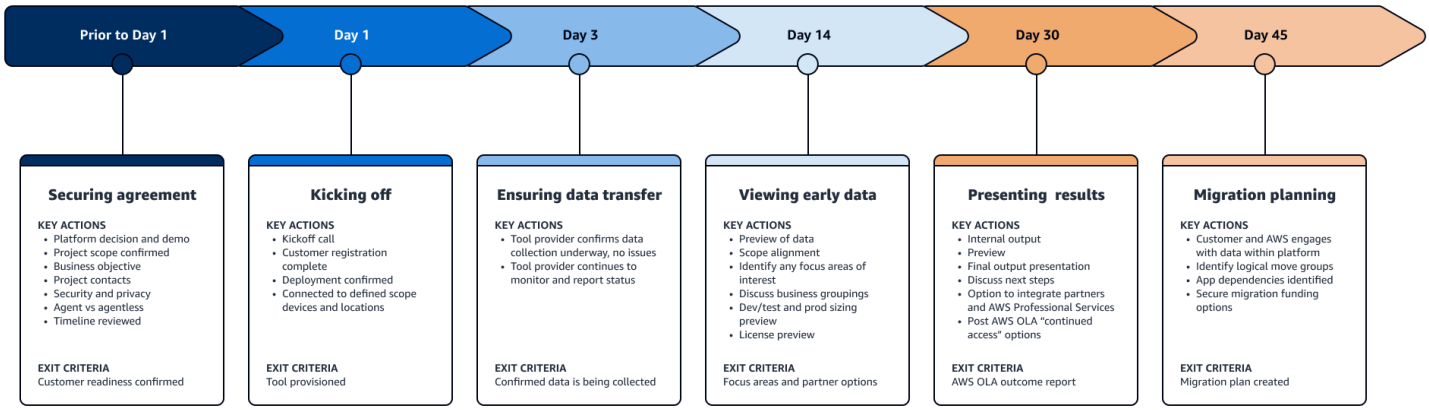
AWS OLA は、次のような提案を行うことで、モダナイゼーションが既存のワークロードに与える影響についてのインサイトも提供します。

- Linux オペレーティングシステムに移動します。
- ARM プロセッサ (AWS Graviton) のアプリケーションサポートを追加します。
- SQL Server ワークロードを Amazon Aurora に移動します。
- Windows と SQL Server のワークロードをオープンソースのテクノロジーに移行することで、ソフトウェア保証を解除します。

次の図は、Windows から Linux への移行や SQL Server から Aurora への移行などのモダナイゼーション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 プロセスには、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、このタイムラインを 1 営業日に短縮できます。さらに、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 AWS できます。

評価の影響

通常、平均的なお客様は、適切なサイジング作業により 20～30% のコスト削減を経験します。適切なサイズ設定は、ソースワークロードを使用状況データに基づいて最適なサイズの AWS インスタンスに一致させます。これらの適切なサイジング調整により、 AWS 環境の月額コストを削減できるだけでなく、多くの場合、組織内の他の場所でコスト削減につながります。例えば、Windows また

は SQL Server のライセンスが 20～30% 増加すれば、Microsoft との次の調整を減らしたり、追加の line-of-business 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適切なサイジングは、最も劇的な財務上の利益を実現する場合によくあります。

AWS は、システムをモダナイゼーション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、金銭的には操作できませんが、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的なトピックから、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。 AWS また、潜在的なイノベーションの機会を検討するのにも役立ちます。

次のステップ

オンプレミス環境または 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、 AWS アカウントチームに連絡して、 AWS OLA. AWS team メンバーに質問に答えて、 AWS OLA が最終的に自分と組織にとって適切な選択であるかどうかの判断を支援してください。または、AWS OLA オンライン をリクエストすることもできます。

追加リソース

- AWS 最適化とライセンスの評価 (AWS ドキュメント)
- AWS re:Invent 2022 - (ENT205) () で AWS コストを節約し、Microsoft ワークロードを最適化する方法 YouTube

Amazon の Windows EC2

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 を使用して、 の安全で信頼性が高く、可用性が高く、適応性の高いインフラストラクチャに Windows Server ワークロードをEC2デプロイ、管理、スケーリングできます AWS クラウド。Amazon で Windows ワークロードを実行することの主な利点を次に示しますEC2。

- スケーラビリティ – Amazon EC2では、要件の変化に合わせて Windows ワークロードを簡単にスケーリングできます。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。実際に使用するリソースに対してのみ料金が発生します。
- 柔軟性 – Windows on Amazon 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EC2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できるため、パフォーマンスを最大化し、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、セキュリティ設定と設定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 – pay-as-you-go 料金モデルを使用すると、使用するリソースに対してのみ料金を支払うことができ、ハードウェアとソフトウェアへの先行投資が不要になります。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。

ガイドのこのセクションでは、以下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
- 適切なサイズの Windows ワークロード
- 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の持ち込み
- Amazon での Windows の支出を最適化する EC2
- 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

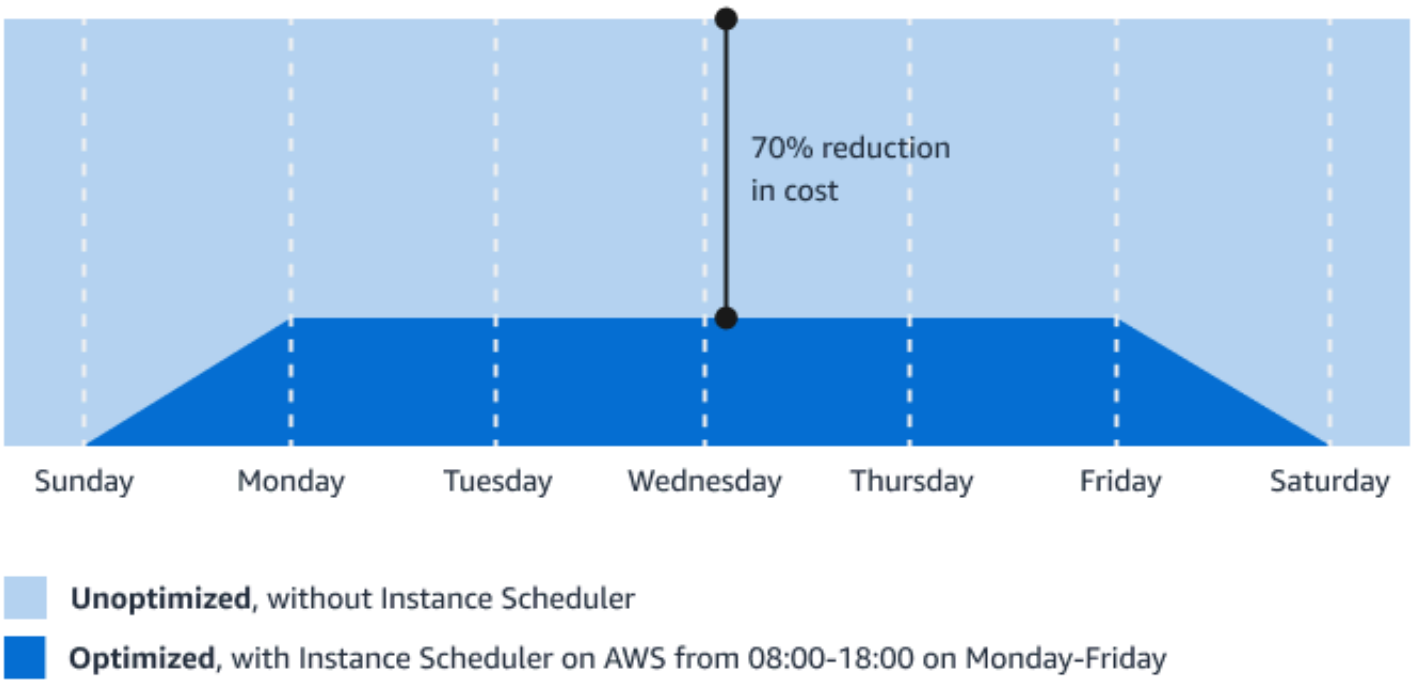
概要

のインスタンススケジューラ AWS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に役立ちます。すべてのインスタンスをフル使用率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する支払い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。

のインスタンススケジューラには、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、および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 も用意されています。これらの機能は、インスタンスをより効果的かつ正確に管理し、さまざまなプロジェクトやチーム間でコストを追跡して割り当てるのに役立ちます。

ケーススタディ

実稼働環境で AWS のインスタンススケジューラを使用して、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稼働状態にしておくと、通常の営業時間内にのみ必要になる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。次の表は、週単位の使用率を 168 時間から 50 時間に削減する方法を示しています。

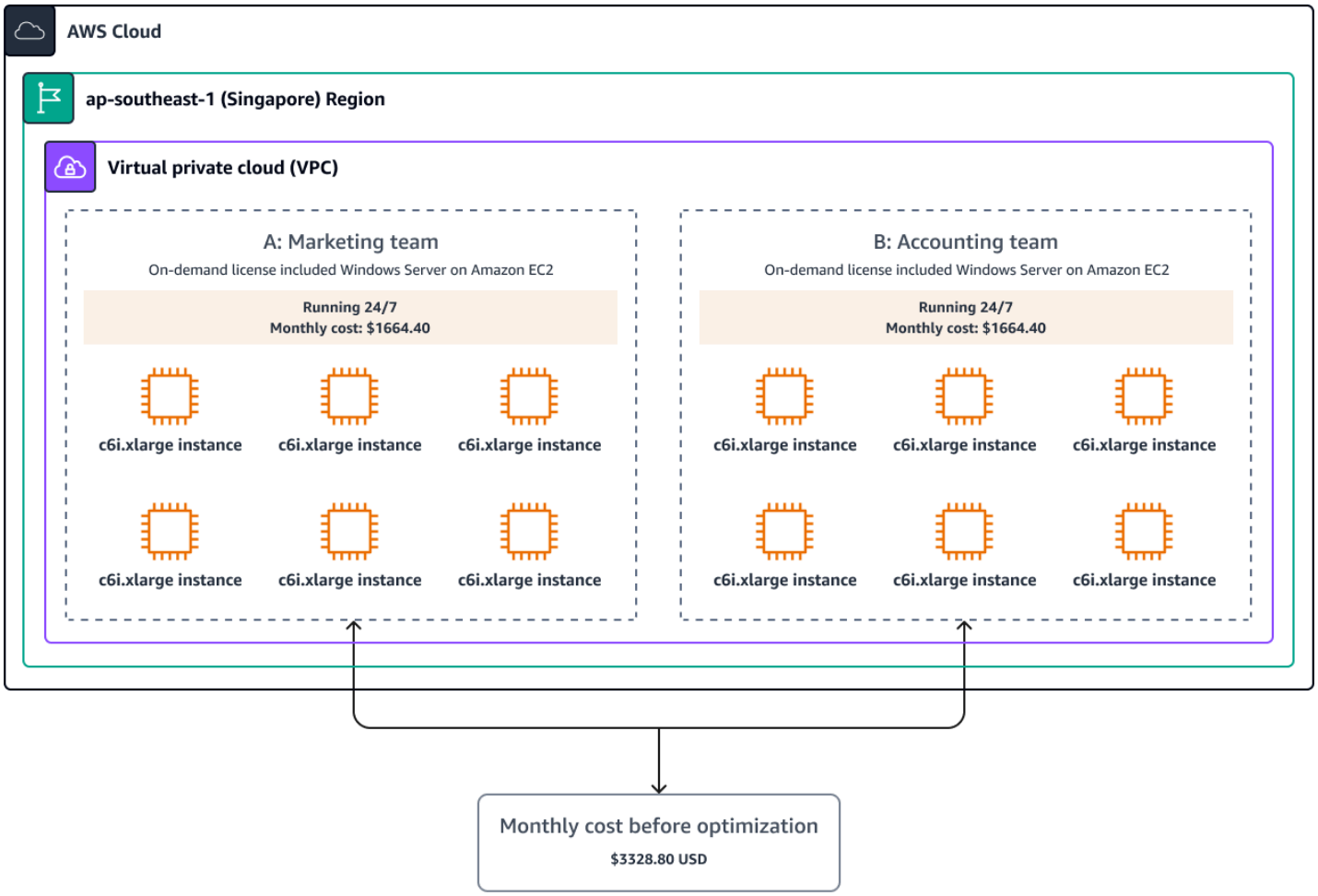


別の例を考えてみましょう。電力会社 Jamaica Public Service Company Limited (JPS) は、データベースを Amazon に移行しましたRDS。現在、 JPSは Amazon EC2 を使用してAPIサービスをホストし、他のアプリケーションを実行します。ではJPS、 のインスタンススケジューラが非本番環境を管理するための主要なツール AWS になりました。JPS は、 でインスタンススケジューラを使用して AWS 開発コストを削減し、チームのニーズと作業スケジュールに基づいてEC2インスタンスを管理しました。これにより、コストが 40% JPS削減されました。詳細については、「Jamaica Public Service が効率的にクラウドに移行し、 AWS インスタンススケジューラ を使用してコストを 40% 削減する」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

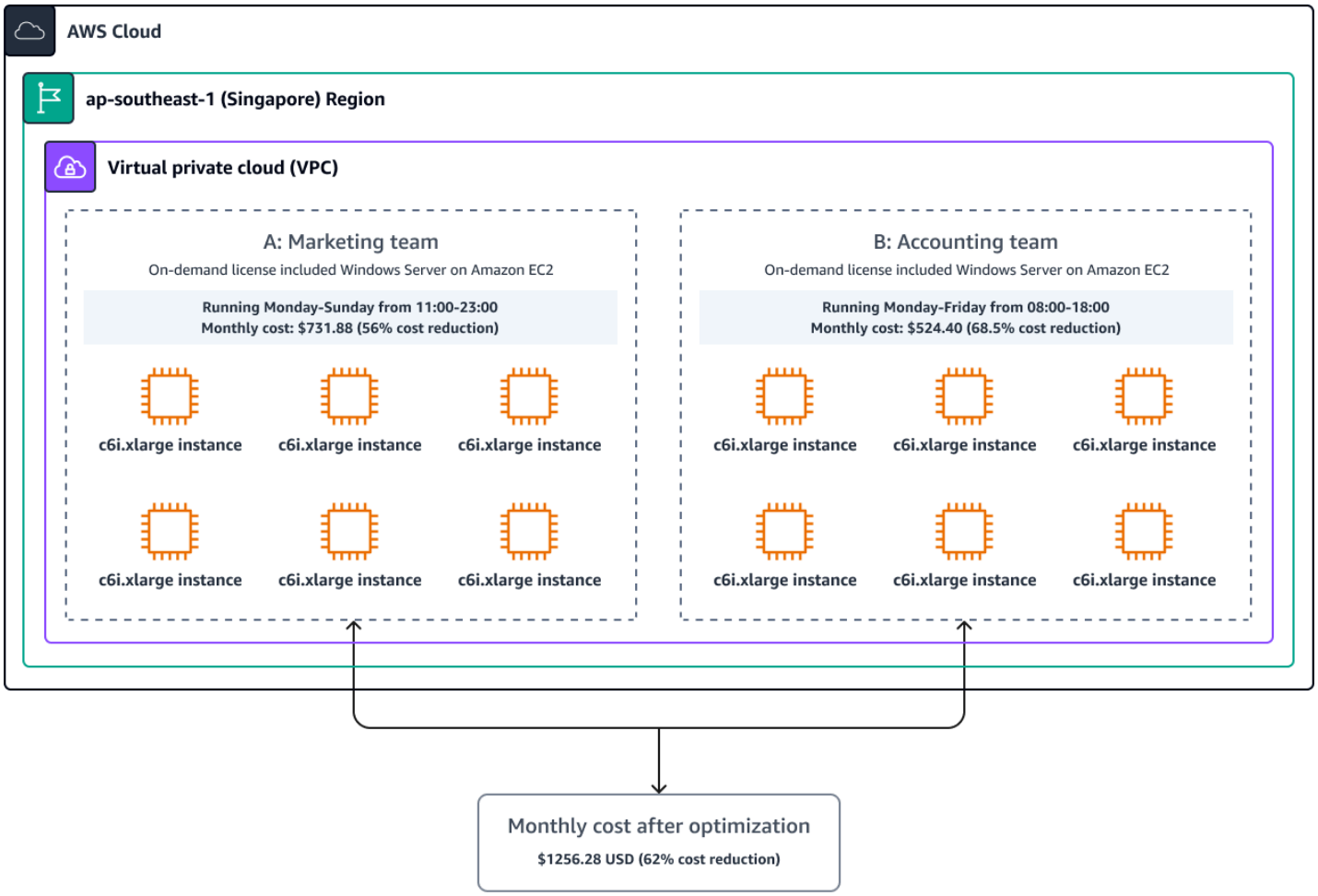
次のシナリオ例は、 でインスタンススケジューラを使用するコスト上の利点を説明するのに役立ちます AWS。このシナリオでは、シンガポールの大手小売企業が Amazon に 2 つの Windows 環境をデプロイしていますEC2。ワークロード A と呼ばれる最初の環境は、マーケティングチームが、店舗が開いている間のリアルタイムの店舗内トランザクションを分析するために使用されます。ワークロード B と呼ばれる 2 番目の環境は、通常の営業時間内にのみ機能する会計チーム用に予約されています。両方の環境の現在の運用スケジュール (24 時間 365 日) は、現在の使用パターンを考慮すると理想的ではなく、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 月の日数は 31 日で、そのうち 23 日が平日です。マーケティングチームがインスタンススケジューラを で使用し AWS 、必要な場合にのみ (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) インスタンスを操作すると、毎月 932.52 ドル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。会計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月 730 時間から 230 時間に短縮されます。これにより、1,140 ドル、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 ドル (62% の削減に相当) 、つまり年間 24,870.24 ドルを節約できます。

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 AWS Pricing Calculator年 3 月に を使用して決定されました。

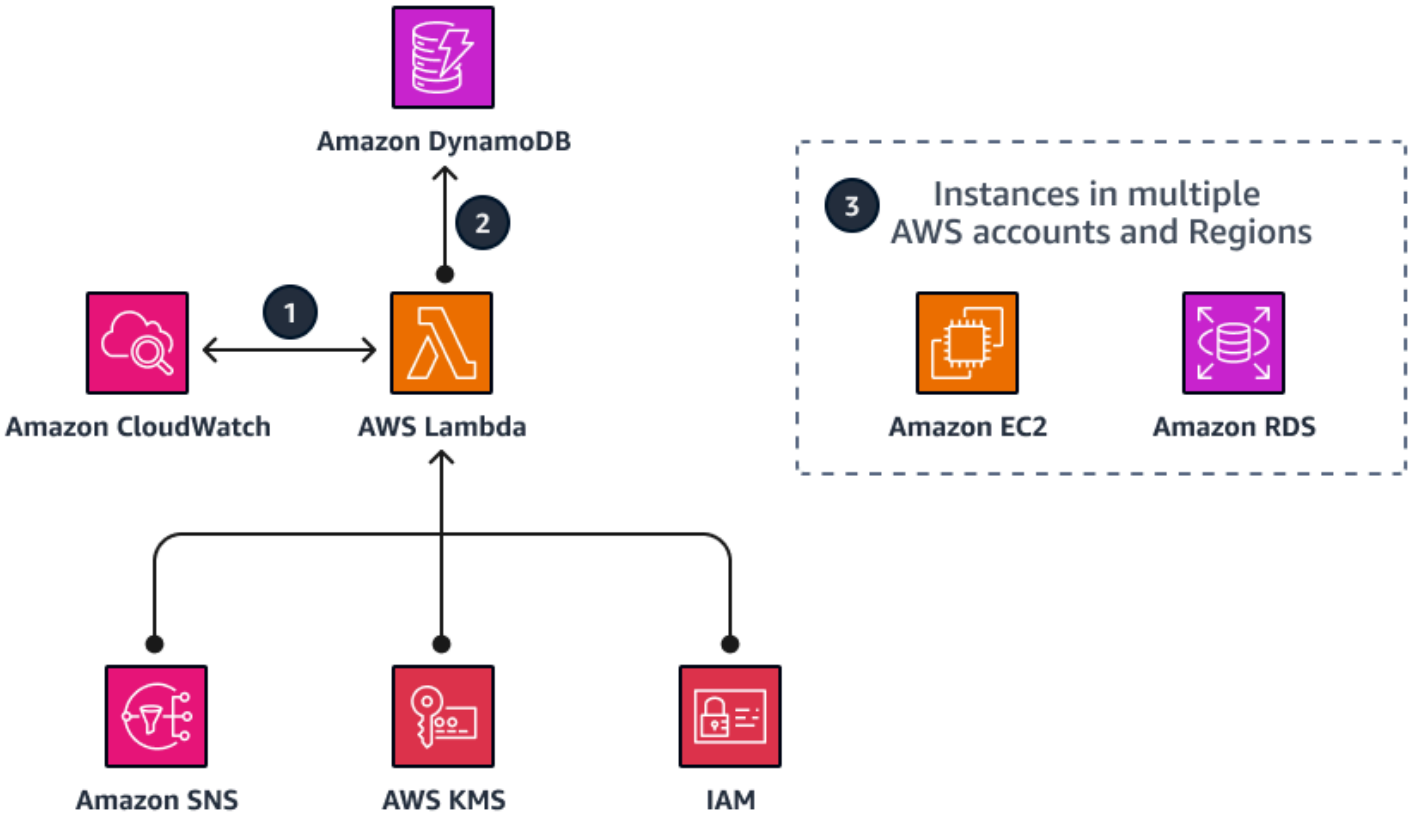
コスト最適化に関する推奨事項

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、 に基づいて AWS インスタンススケジューラをデプロイおよび設定する方法について説明します。でインスタンススケジューラ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. インスタンススケジューラ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、インスタンススケジューラスタック AWS クラウド によって で作成された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、 のインスタンススケジューラがスケジュールを該当する Amazon EC2インスタンス、Amazon RDSインスタンス、クラスターに関連付けるために 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 関数は実行のたびに取得します。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. インスタンススケジューラの初期設定時に、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、指定した名前が、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。

インスタンススケジューラスタックを起動する

このセクションでは、 でインスタンススケジューラの CloudFormation スタックを起動する方法について説明します AWS。

> **Note**
>
> でインスタンススケジューラを実行している間 AWS のサービス に使用される のコストは、お客様の負担となります AWS。2023 年 1 月現在、このソリューションを us-east-1リージョンのデフォルト設定で実行するためのコストは、Lambda の料金で月額約 9.90 USD です。Lambda 無料利用枠の月額利用クレジットがある場合はそれ以下です。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「コスト」セクションを参照してください。

インスタンススケジューラスタックを起動するには、次の手順を実行します。

1. にサインインし、起動ソリューション (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AWS Management Console。

   > **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。別のリージョンでインスタンススケジューラを起動するには、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ます。

3. スタックの作成ページで、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あることを確認し、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。名前文字の制限の詳細については、 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 と呼ばれますMyInstanceScheduler。

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

6. UnderParameters 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. [Next (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、設定を確認して確認します。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 DynamoDB テーブルが作成されます。このテーブルには、独自のカスタム期間ルールとスケジュールを作成するためのリファレンスとして使用できるサンプル期間ルールとスケジュールが含まれています。期間設定の例については、「 ドキュメント」の「インスタンススケジューラのサンプルスケジュール AWS 」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:

```
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、次の手順を実行します。

1. DynamoDB コンソールにサインインし、 でインスタンススケジューラの CloudFormation テンプレートを起動したのと同じリージョンにいることを確認します AWS。

2. ナビゲーションペインで、テーブル を選択し、 という名前のテーブルを選択しますConfigTable。
3. Explore table items を選択します。
4. 営業時間の期間を作成するには、営業時間項目の期間を選択します。
5. 項目の編集ページで、開始時刻を 0800、終了時刻を 1800 に変更します。平日はデフォルト値のままにします。

> **Note**
>
> 開始時刻と終了時刻の値は、インスタンスを開始および停止するタイミングを決定し、平日の値は、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. オフィス時間期間を複製し、それを使用して小売時間の新しい期間を作成するには、オフィス時間項目の期間を選択します。次に、アクションメニューから、重複項目 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 でConfigTable、項目リストにリストされている、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

のインスタンススケジューラのコンテキストでは AWS、スケジュールは 1 つ以上の期間と関連するタイムゾーンの適用を指します。その後、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、前のセクションで作成した期間にスケジュールを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、次の手順を実行します。

1. DynamoDB コンソールにサインインし、 でインスタンススケジューラの CloudFormation テンプレートを起動したのと同じリージョンにいることを確認します AWS。
2. ナビゲーションペインで、テーブル を選択し、 という名前のテーブルを選択しますConfigTable。
3. Explore table items 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の新しいスケジュール (この例ではシンガポールの営業時間) を作成するには、uk-office-hours 項目のスケジュールを選択します。次に、アクションメニューから、重複項目 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 を繰り返して、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 でConfigTable、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。内のタグエディタを使用してAWS Resource Groups、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. にサインインAWS Management Consoleし、 CloudFormation テンプレートを以前に起動したのと同じリージョンにいることを確認します。
2. Resource Groups 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. タグを付けるリソースの検索セクションで、リージョン でリージョンを選択します。リソースタイプ では、Amazon EC2または Amazon を選択しますRDS。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、部門キーとマーケティング値が既に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の管理 を選択します。
6. 選択したすべてのリソースのタグの編集セクションで、タグの追加を選択してインスタンススケジューラのスケジュールタグをEC2インスタンスに追加します。schedulea に一致するタグキーと値を使用できます (以前は DynamoDB で作成されています) 。
7. タグキー には、スケジュール を追加します。タグ値 には、 と入力しますsingapore-retail-hours。
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、選択したすべての 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

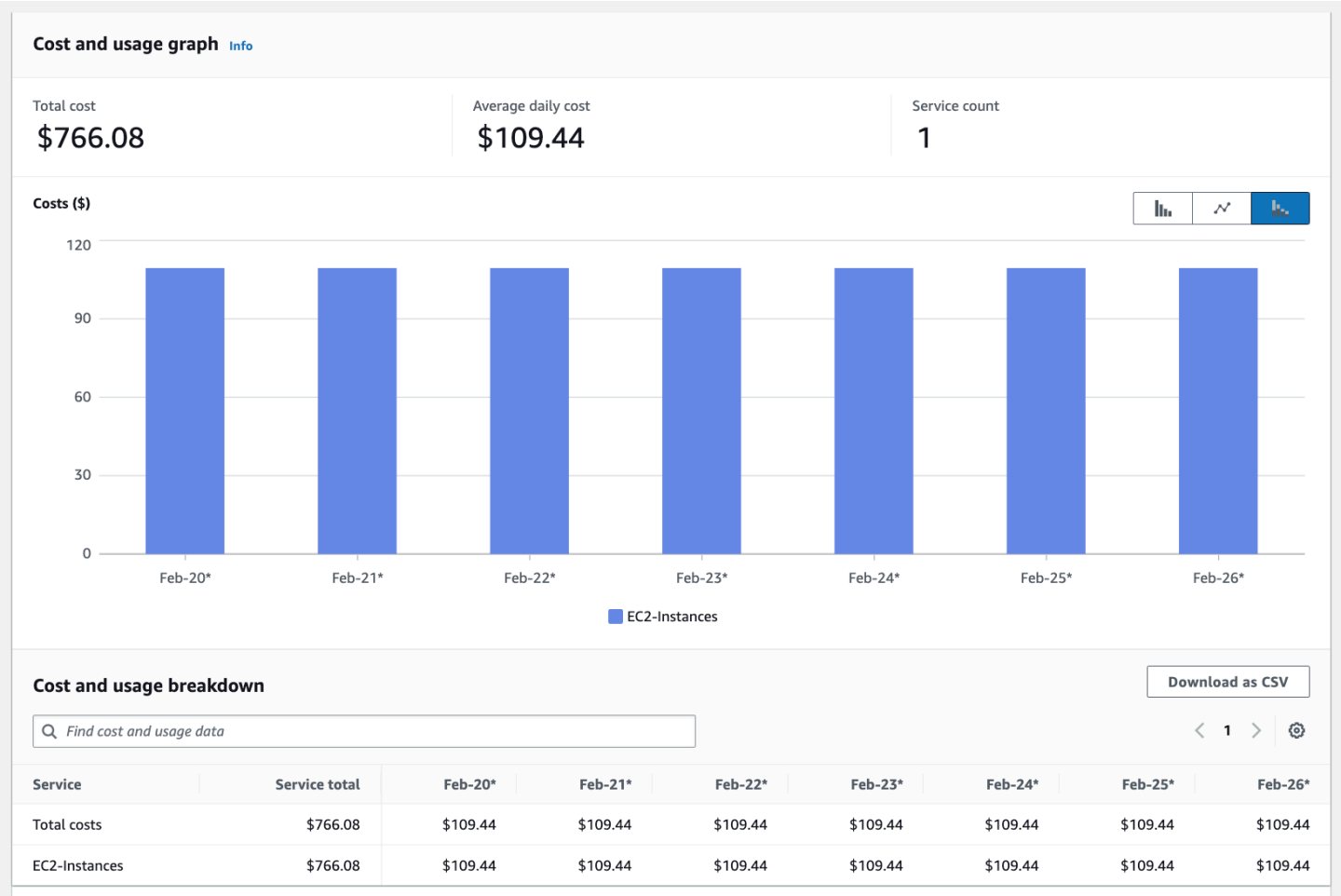
でインスタンススケジューラを使用するコストメリットを測定するAWS Cost Explorerために を使用することをお勧めします AWS。Cost Explorer を使用して、以下を実行できます。

- Instance Scheduler によって管理されるEC2インスタンスなど、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。

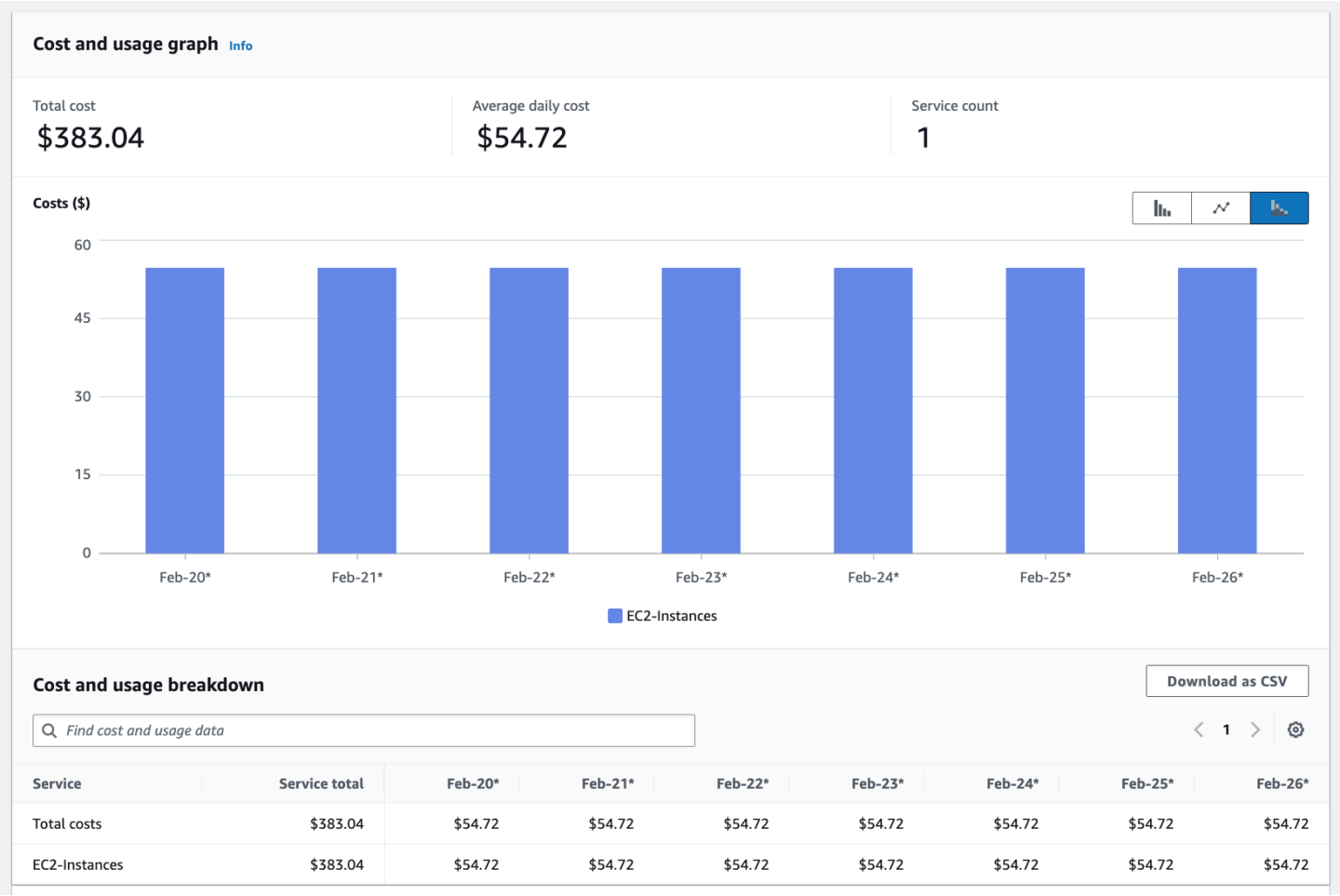
- Cost Explorer ビューをタグでフィルタリングして、特定のワークロードに集中し、インスタンススケジューラを使用して達成されたコスト削減を詳細に把握できるようにします。
- インスタンススケジューラの使用による財務上の影響に関するインサイトを取得します。
- さらなるコスト最適化の機会を特定し、 AWS 支出を最適化するためのデータ主導型の意思決定を行います。

次のグラフは、Instance Scheduler を使用して最適化前の 7 日間 (月曜日から日曜日) にワークロード A とワークロード B を運用した場合のコストを示しています。

ワークロード A と B の合計コスト

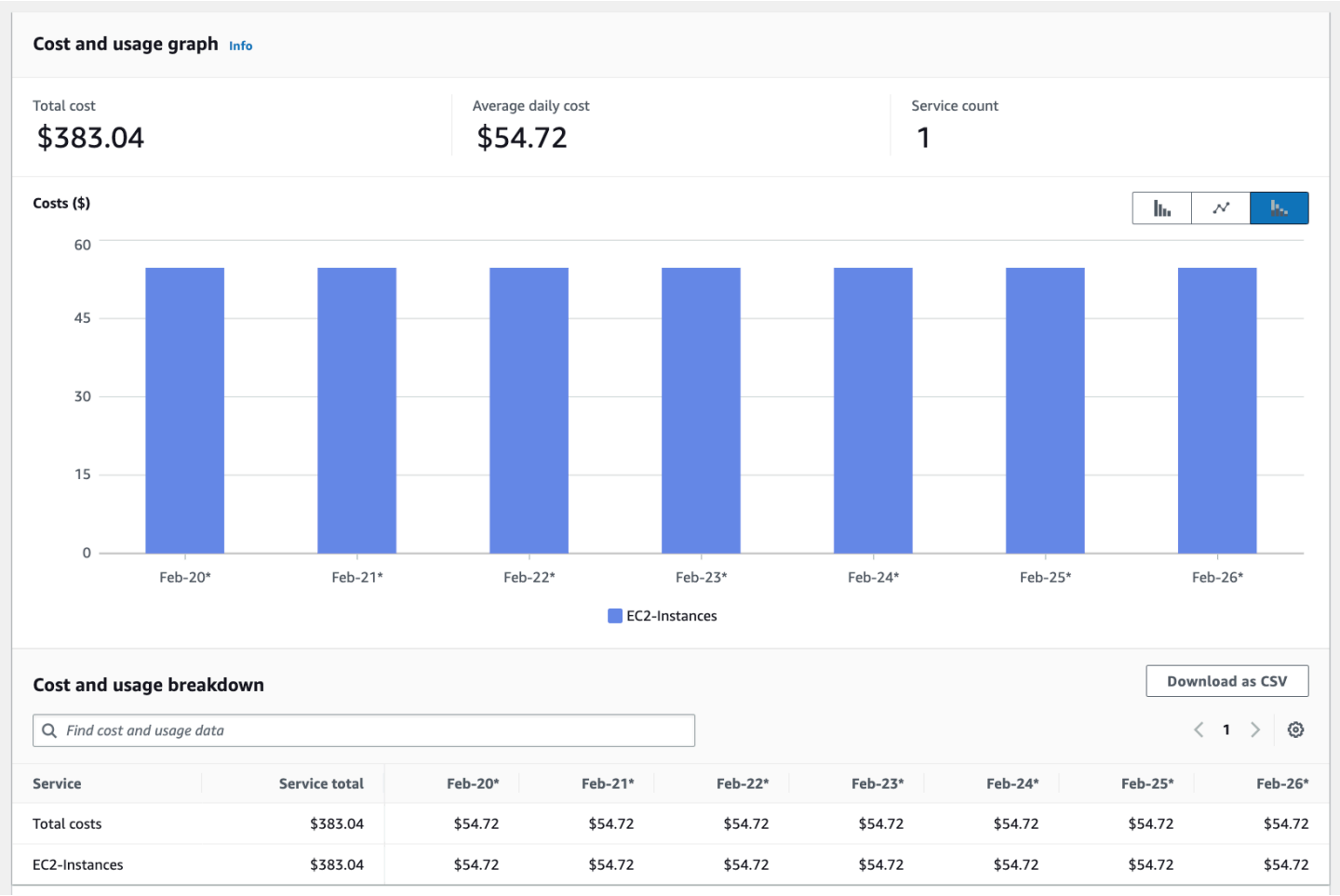


ワークロード A の経費



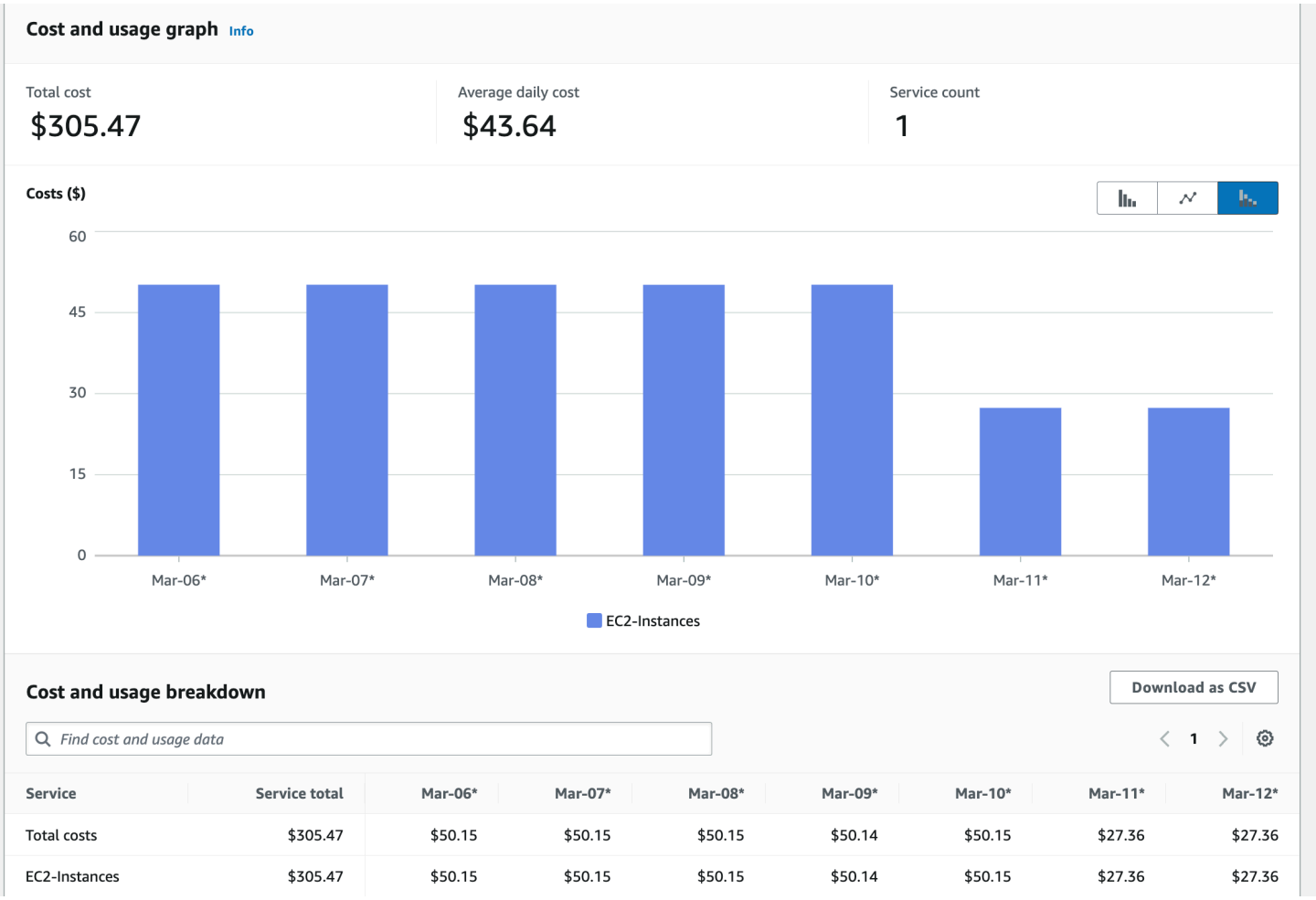
Service	Service total	Feb-20*	Feb-21*	Feb-22*	Feb-23*	Feb-24*	Feb-25*	Feb-26*
Total costs	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72
EC2-Instances	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72

ワークロード B の経費



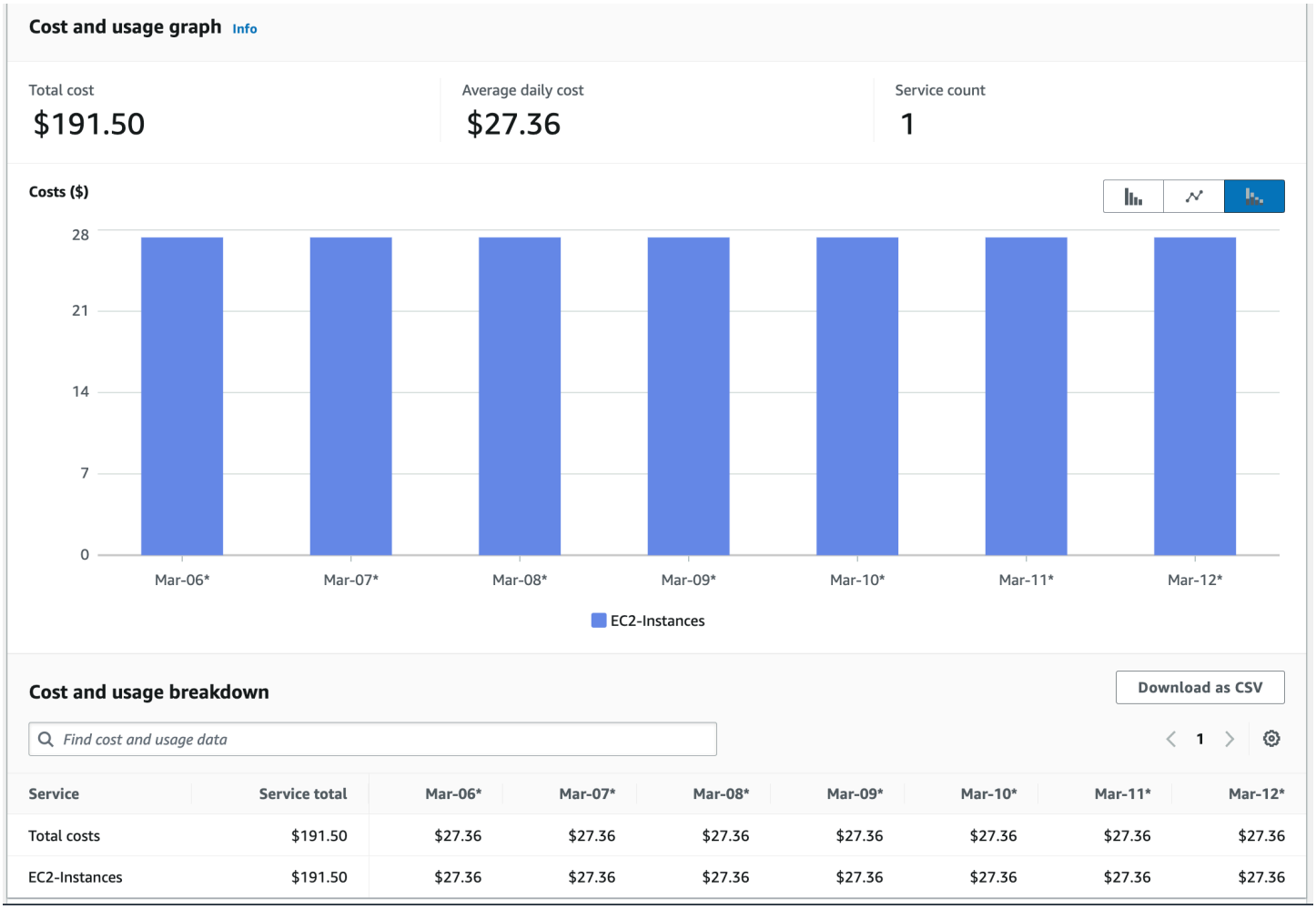
このシナリオでは、Cost Explorer は にインスタンススケジューラを実装した結果のコスト削減を示します AWS。次のグラフは、最適化後の 7 日間 (月曜日から日曜日) のワークロード A とワークロード B の運用コストを示しています。

ワークロード A と B の合計費用

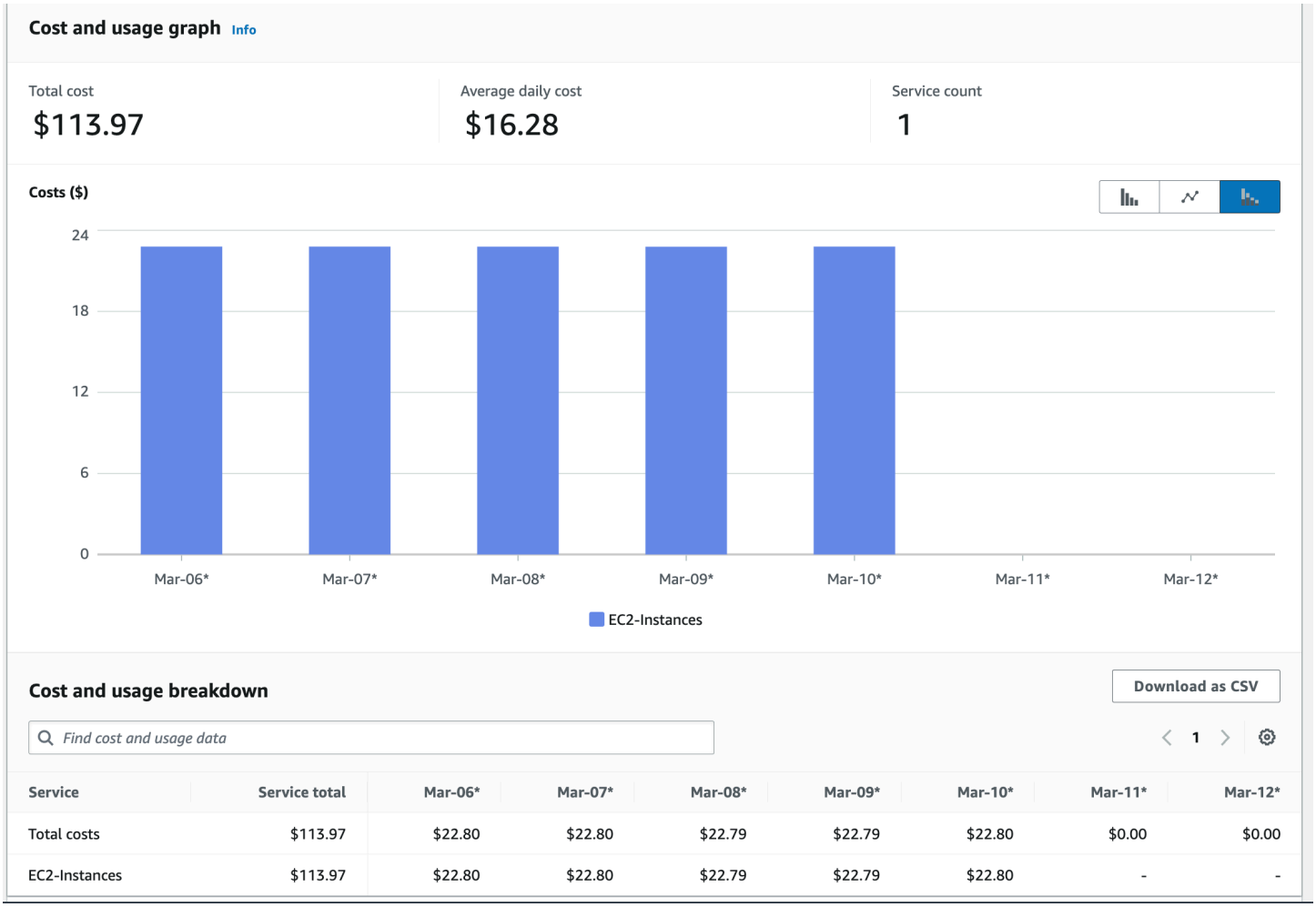


Service	Service total	Mar-06*	Mar-07*	Mar-08*	Mar-09*	Mar-10*	Mar-11*	Mar-12*
Total costs	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36
EC2-Instances	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36

ワークロード A の経費


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 の経費



追加リソース

- AWS インスタンスの開始と停止を自動化する (AWS ドキュメントのインスタンススケジューラ)
- 基本に戻る: インスタンススケジューラを使用して Amazon EC2および Amazon RDSリソースコストを制御する (YouTube)
- AWS リソースのタグ付け (Tagging AWS Resources ユーザーガイド)
- (ドキュメント) を使用したコストの分析 AWS Cost ExplorerAWS Billing and Cost Management

適切なサイズの Windows ワークロード

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。

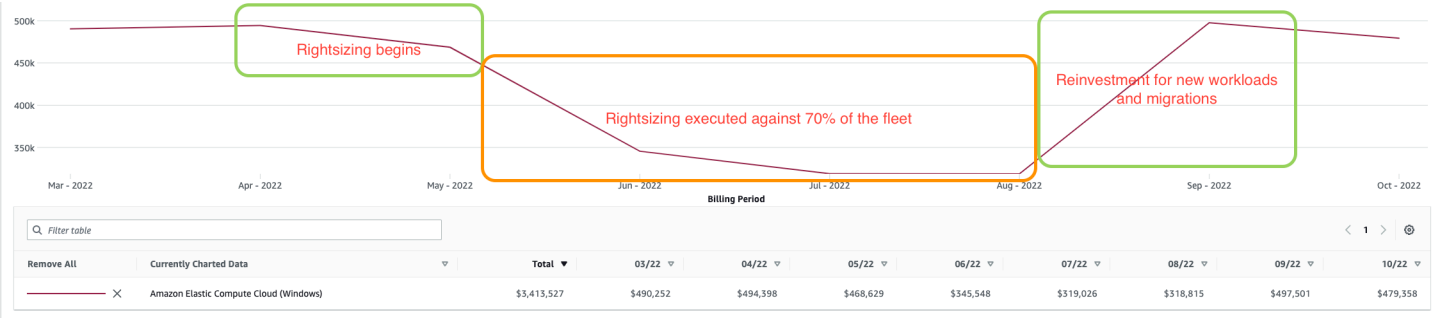
このセクションでは、 AWS Compute Optimizerを使用して Amazon EC2の適切なサイジングの機会を特定する方法について説明します。Compute Optimizer は、次のタイプの AWS リソースの過剰プロビジョニングと過少プロビジョニングを防ぐのに役立ちます。

- 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
- 上の Amazon Elastic Container Service (Amazon ECS) サービス AWS Fargate
- AWS Lambda Amazon CloudWatch が提供する使用率データに基づく 関数

コスト最適化シナリオ

適切なサイジングの効果を測定するのは難しい場合があります。これは、適切なサイジングの取り組みを特定のアプリケーション、チーム、または組織全体に向けることができるためです。例えば、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、アカウントと 全体で大幅なオーバープロビジョニングを検出できます AWS リージョン。その後、AWS Systems Manager オートメーションを使用して、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正化できます。その結果、組織はフリートの 70% で適切なサイズのインスタンスタイプを調整し、35% のコスト削減を達成しています。

次のダッシュボードは、この例の組織が Compute Optimizer の適切なサイジング推奨事項を戦略的に実装したため、数か月にわたって達成された削減を示しています。その目的は、契約の終了に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できるだけ効率的に運用することでした。



コスト最適化に関する推奨事項

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 のレコメンデーションを使用する
- 適切なサイジングのためのインスタンスのタグ付け
- コスト配分タグが AWS 請求ツールと連携できるようにする
- Automation を使用して適切なサイジングの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 でコストの前後に確認する

Compute Optimizer を有効にする

Compute Optimizer は、組織または の単一アカウントレベルで有効にできます AWS Organizations。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で、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、適切なサイズをアクティビティではなく繰り返しの point-in-time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が組織を可視化し、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。これを組織レベルで有効にするには、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 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を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すると、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、そのレコメンデーションを参照できます。

> **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、最小特権の原則を行使できます。これにより、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、 にアクセスできない場合は AWS Organizations、そのアカウントとリージョンの 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「 の開始方法 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、Compute Optimizer に、組織内で十分な情報に基づいた適切なサイジングのレコメンデーションを行うために必要な必須メトリクスを提供します。これは、レコメンデーションを提供する前に 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。でメモリメトリクスを使用する場合、追加料金は発生しません CloudWatch。

CloudWatch エージェントを有効にし、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 （ IAM) の AWS マネージドポリシー AWS Systems Managerがアタッチされたロール – これは Systems Manager Automation ドキュメント用です。
- AWS Systems Manager ドキュメント – エージェントをインストールして設定します CloudWatch (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、インスタンス上の既存の 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 Management Console 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. [Next (次へ)] を選択します。
5. テンプレートリソース 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、`ComputeOptimize.yml`ファイルをアップロードします。
7. [Next (次へ)] を選択します。
8. スタックの詳細を指定ページで、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. リソースの特定ページで、インポートするリソースの識別子値を入力します。
10. リソースのインポート を選択します。
11. スタックがデプロイされたら、Outputs 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のノード管理セクションで、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。

でメトリクスを表示する CloudWatch

メトリクスが 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 CloudWatch。

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 の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を考えてみましょう。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。適切なサイジング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によって正確に実行されます。

組織の管理アカウント内から Compute Optimizer に移動した場合 (次のステップに示すように)、調査するアカウントを選択できます。この例では、us-east-1リージョン内の 1 つのアカウントで 6 つのインスタンスが実行されています。6 つのインスタンスすべてがオーバー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、レコメンデーションの詳細をエクスポートする

1. にサインイン AWS Management Console 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ます。
3. ダッシュボードページの検索ボックスに、リージョン=米国東部 (バージニア北部) と入力します。次に、Findings=Over-provisioned と入力します。これらのフィルターを使用すると、us-east-1リージョン内のすべてのオーバープロビジョニングされたインスタンスを表示できます。
4. オーバープロビジョニングされた EC2インスタンスの詳細なレコメンデーションを確認するには、EC2インスタンスカードまで スクロールダウンし、レコメンデーションの表示 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送信先とする 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レコメンデーションを保存するには、Compute Optimizer が各リージョンで に書き込むことができる 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「 の Amazon S3 バケットポリシー AWS Compute Optimizer」を参照してください。

7. フィルターの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レコメンデーションを含めるチェックボックスをオンにします。
8. リソースタイプ では、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オンにします。
10. [エクスポート] をクリックします。

レコメンデーションに基づいてインスタンスを選択する

インスタンスのレコメンデーションは、Compute Optimizer によって収集および分析される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 R6iR5、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も低くなります。R5 インスタンスと R6 インスタンスのコストは 1 時間あたり同じで、ほぼ同じです。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 プロセッサーを搭載し、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 は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インスタンスのレコメンデーションを選択します。このページでは、現在のインスタンスタイプを置き換える推奨オプションと比較します。
2. 適切なサイズのインスタンスの ID を取得するには、 の管理アカウントから Amazon S3 コンソールを開きます AWS Organizations。
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 Text to Columns ボタンを使用します。

> **Note**
>
> インスタンスIDsは Amazon リソースネーム () として表されますARNs。必ず区切り文字を「/」に設定し、インスタンス ID を抽出します。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 をトリミングすることもできますARN。

6. Excel では、検出結果列をフィルタリングして OVER_PROVISIONED インスタンスのみを表示します。これらは、適切なサイジングをターゲットにしてい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、インスタンスをテキストエディタIDsに保存します。

適切なサイジングのためのインスタンスのタグ付け

ワークロードのタグ付けは、 でリソースを整理するための強力なツールです AWS。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、メンテナンスウィンドウ中にサイズ変更の対象となるオーバープロビジョニングされたインスタンス全体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 Management Console し、サイズ変更対象の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 では、ターゲットリージョンを選択します。
4. リソースタイプ では、 を選択します AWS::EC2::Instance。
5. [リソースを検索] を選択します。
6. リソース検索結果ページで、適切なサイズにするすべてのインスタンスを選択し、選択したリソースのタグの管理 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、「ライツサイジング」と入力します。タグ値 には、有効な を入力します。次に、「レビュー」を選択し、タグの変更を適用します。

> **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ると、後で Cost Explorer の でフィルタリングできます。

ユーザー定義タグを作成してリソースに適用した後、アクティベーション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 24 時間かかる場合があります。アクティベーションにタグを選択すると、タグがアクティブになるまでにさらに 24 時間かかる場合があります。

上級ユーザーの場合、ターゲットアカウントとリージョンAWS CloudShell内で を使用して、複数のインスタンスにタグ付けできます。例:

```
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の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 タグを認識して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と、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については、ドキュメントの AWS Billing and Cost Management 「ユーザー定義のコスト配分タグの有効化」を参照してください。

1. にサインイン AWS Management Console 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、「ライツサイジング」と入力します。
4. Rightsizing タグキーを選択し、「 のアクティブ化」を選択します。

24 時間後、 タグは Cost Explorer に表示されます。

Systems Manager Automation を使用して適切なサイジングのレコメンデーションを実装する

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、この中断をメンテナンスウィンドウで処理する必要があり、独自のサイズ変更を処理するために異なる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ドキュメントの互換性のあるインスタンスタイプの考慮事項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、アカウントとリージョンごとに適切なサイジングのレコメンデーションを実装します。このアプローチは、ほとんどの組織で一般的です。ほとんどの組織では、目的に応じて異なるインスタンスタイプが必要です。また、同じAWS-ResizeInstanceオートメーションドキュメントを使用して、単一アカウントとマルチアカウントのデプロイをターゲットにすることもできます。

1. にサインイン AWS Management Console 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. [オートメーションを実行] を選択します。
5. オートメーションランブックの実行ページで、シンプル実行 を選択します。
6. 入力パラメータ セクションで、 InstanceIdと を入力しますInstanceType。残りのデフォルト値は保持します。
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経るのを 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。

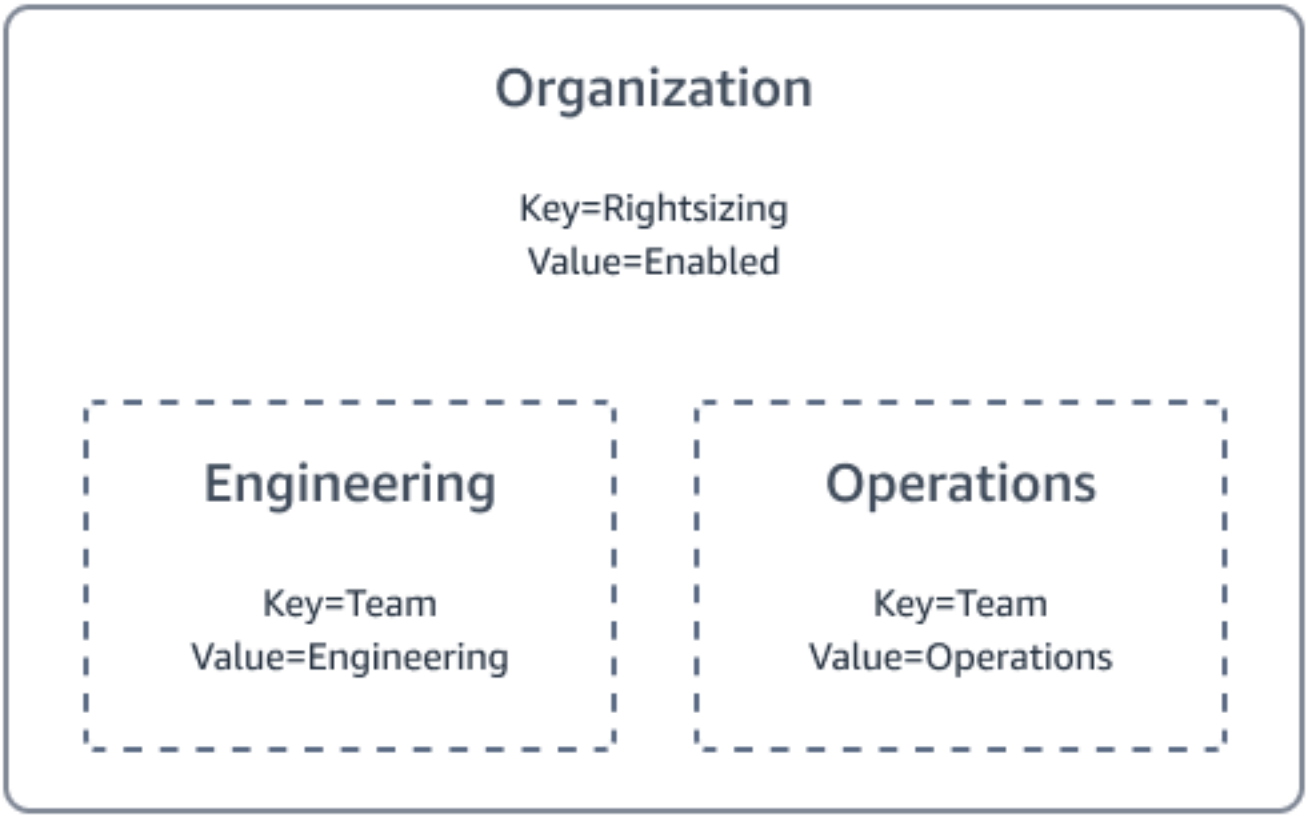
複数のアカウントとリージョンで適切なサイジングプロセスを使用する場合は、カスタム Systems Manager Automation ドキュメントを作成する必要があります。このドキュメントでは、複数のインスタンスをパラメータとしてフィードし、ターゲットインスタンスを同じ送信先インスタンスタイプ

に移動できます (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 でコストの前後に確認する

リソースのサイズが適切になったら、Cost Explorer を使用して、Rightizing タグを使用してコストの前後に表示することができます。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。このガイドで説明している例では、ライツサイジングタグを使用して、汎用タグをすべてのターゲットインスタンスに適用します。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。



オペレーションチームが所有している本番稼働用ウェブサーバーのサイズを適切に調整するビジネスの例を考えてみましょう。Cost Explorer では、Rightizing タグは有効な に設定され、チームタグはオペレーション に設定されます。この例では、適切なサイジングにより、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、適切なサイジング前の年間コストは 7,945.92 USD です。適切なサイズ変更後、年間コストは 2,499.84 ドル

に低下します。これにより、年間ワークロードコストが 68.5% 削減されます。これを大規模な組織全体に与えた影響を想像してください。これはサンプル環境で行われ、インスタンスはほぼアイドル状態であることに注意してください。本稼働環境では、10～35% の節約が可能です。

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サイズを適切に調整することの影響について考えてみましょう。Cost Explorer では、Rightizing タグは有効 に設定され、チームタグはエンジニアリング に設定されます。この例では、適切なサイジングにより、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、適切なサイジング前の年間コストは 6,696.00 USD です。適切なサイズ変更後、年間コストは 3,928.32 ドルに低下します。

複数のタグを使用する場合は、データを詳細なコスト詳細に絞り込むことができます。この例では、チームタグはノイズを減らし、チームレベルで影響を表示できます。Rightsizing タグが有効になっているため、そのタグを持つインスタンスを の値が有効であるか、値が存在しない場合に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 (支払い者) で表示する場合に、適切なサイジングの取り組みをグローバルに表示できます。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。

単一のアカウントレベルで、Rightizing タグが有効な に設定されている例を考えてみましょう。 運用コストは 1 時間あたり 1.64 USD から 1 時間あたり 0.72 USD に低下します。1 か月あたり 744 時間と仮定すると、適切なサイジング前の年間コストは 14,641.92 USD です。適切なサイズ変更後、年間コストは 6,428.16 ドルに低下します。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。

適切なサイジング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、 に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。には、適切なサイズに関する推奨事項とライセンスガイダンス AWS OLAも用意されています。
- Savings Plans を購入する前に、適切なサイズ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 を超える購入を回避できます。

レコメンデーション

次のステップを実行することをお勧めします。

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

2. Savings Plansを確認します。

追加リソース

- 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
- 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 ブログの Microsoft Workloads)

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

オンプレミス環境と比較してクラウドで運用されているワークロードの大きな違いは、オーバープロビジョニングの実践です。オンプレミス用に物理ハードウェアを購入する場合、設備投資は、通常 3 ～ 5 年という所定の期間にわたって継続することが予想されます。ハードウェアの存続期間中に予想される成長に対応するため、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、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ることがよくあります。

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活用する効果的な手段として浮上しました。管理者は vCPUs と VMsでオーバープロビジョニングされているためRAM、ハイパーバイザーは、未使用のリソースを各 VM に割り当てることで、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量を管理できます。を管理する場合VMs、各 VM に割り当てられた vCPU 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ました。VM リソースの過剰割り当ては、利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VMsでの過剰プロビジョニングを控えます。これは不要です。クラウドコンピューティングは運用コストであり、設備投資ではなく、使用した分に対してのみ支払います。ワークロードに将来的により多くのリソースが必要な場合は、事前に必要になるのではなく、実際に必要になったときにプロビジョニングしてください。

適切な Amazon EC2インスタンスタイプ を選択するには、数百のオプションがあります。Windows ワークロードをクラウドに移行する予定の場合、 は現在のワークロードAWS OLAをよりよく理解し、 でのパフォーマンスの例を提供するために AWS を提供します AWS。この 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に合わせることを目的としています。

すでに Amazon でワークロードを実行EC2していて、コスト最適化戦略を探求している場合は、このガイドのこのセクションは、Amazon EC2インスタンスと一般的な Windows ワークロードへの適用性の違いを特定するのに役立ちます。

コスト最適化に関する推奨事項

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価格変動を理解する
- EC2 世代間の価格とパフォーマンスの違いを理解する
- 新しいインスタンスへの移行
- バースト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、次のカテゴリに分類されます。

- 汎用
- コンピューティングの最適化

- メモリ最適化

これをさらに簡素化するには、各カテゴリのベースラインEC2インスタンスを検討してください。

- コンピューティング最適化 – C6i
- 汎用 – 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 の量により大きくなりましたRAM。インスタンスの数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	vCPUs	RAM	時間料金
c6i.xlarge	4	8	$0.17
m6i.xlarge	4	16	$0.19
r6i.xlarge	4	32	0.25 ドル

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

バーストインスタンス

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料金を回避するのがベストプラクティスですが、必要な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、24 時間アクセス可能である必要があります。

バーストインスタンス (T3) は、コンピューティングコストを低く抑えながら、急速なワークロードや使用率の低いワークロードを 1 日中オンラインに維持する方法を提供します。バーストEC2インスタンスには、インスタンスが短期間で使用できる 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、バーストクレジット に基づくシステムを使用しますCPU。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。バーストインスタンスはさまざまな vCPU-to-RAM比率を提供し、場合によっては最適化されたインスタンスを計算し、他の場合は他の汎用インスタンスを計算します。

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 (バーストインスタンス) の時間単位の料金を示しています。

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

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

プロセッサアーキテクチャ間の価格変動を理解する

インテルプロセッサは、インスタンスの開始以来、EC2インスタンスの標準となっています。C5, M5R5 などの以前の世代のEC2インスタンスは、Intel をプロセッサアーキテクチャ (デフォルト) として示していません。C6i, M6iR6i などの新しい世代のEC2インスタンスには、Intel プロセッサーの使用を示す「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 と最も比較可能なプロセッサは AMD (「a」で表されます) です。AMD EPYC プロセッサは同じ x86 アーキテクチャを使用し、Intel プロセッサに似たパフォーマンスを低価格で提供します。次の料金例に示すように、AMDEC2インスタンスは Intel と比較してコンピューティングコストを約 10% 削減します。

Intel インスタンス	時間料金	AMD インスタンス	価格	% の差
c6i.xlarge	$0.17	c6a.xlarge	$0.153	10%
m6i.xlarge	$0.192	m6a.xlarge	$0.1728	10%
r6i.xlarge	$0.252	r6a.xlarge	$0.2268	10%

> **Note**
>
> 料金は、 `us-east-1` リージョンのオンデマンド時間単位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 (「g」で示されます) です。によって設計された Graviton プロセッサは AWS、Amazon で最高の価格パフォーマンスを提供しますEC2。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、20% 以上のパフォーマンス向上も実現します。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と予想され、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ります。

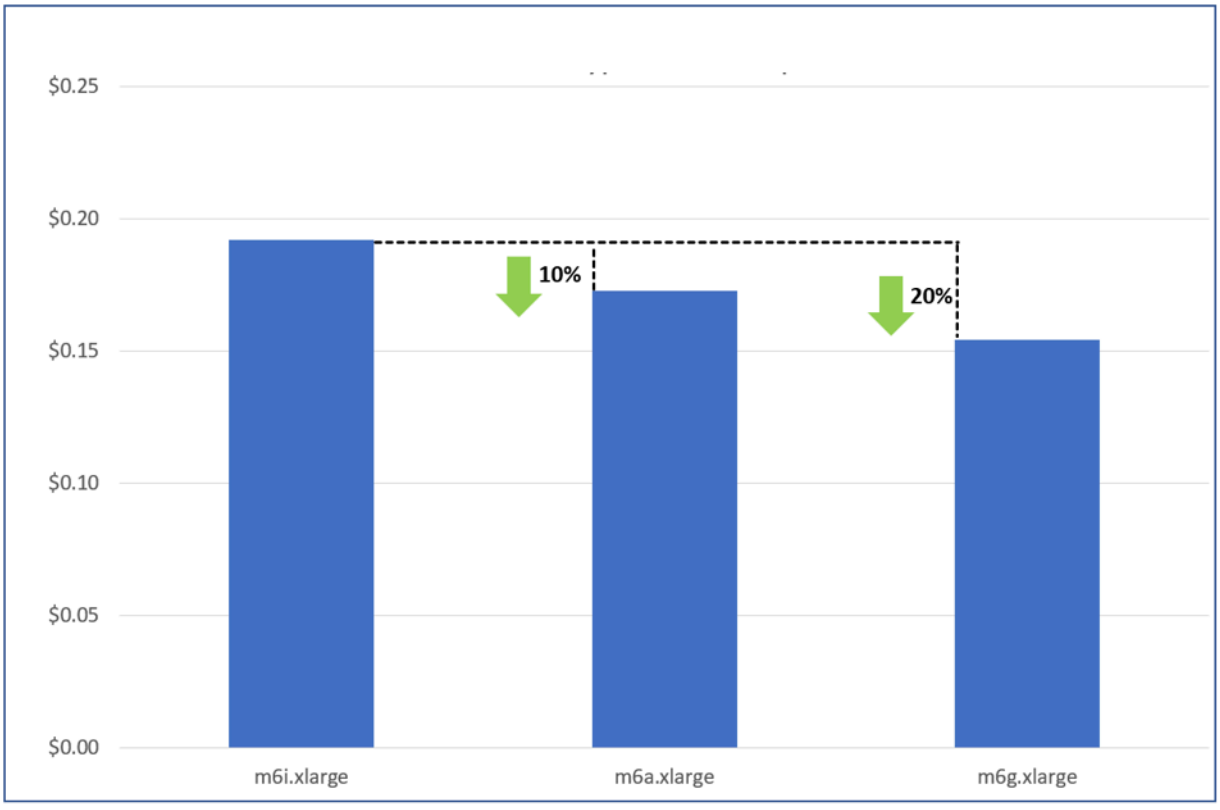
Windows Server は、ARMアーキテクチャに基づく Graviton プロセッサでは実行できません。実際、Windows Server は x86 プロセッサでのみ動作します。Windows Server に Graviton ベースのインスタンスを使用して 40% の料金パフォーマンスを向上させることはできませんが、特定の Microsoft ワークロードで Graviton プロセッサを引き続き使用できます。例えば、新しいバージョンの 。NET は Linux で実行できます。つまり、これらのワークロードはARMプロセッサを使用し、より高速で手頃な価格の Graviton EC2インスタンスから恩恵を受けることができます。

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	% の差
c6i.xlarge	$0.17	c6g.xlarge	$0.136	20%
m6i.xlarge	$0.192	m6g.xlarge	$0.154	20%
r6i.xlarge	$0.252	r6g.xlarge	0.2016 ドル	20%

> **Note**
>
> 料金は、 `us-east-1` リージョンのオンデマンド時間単位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ています。



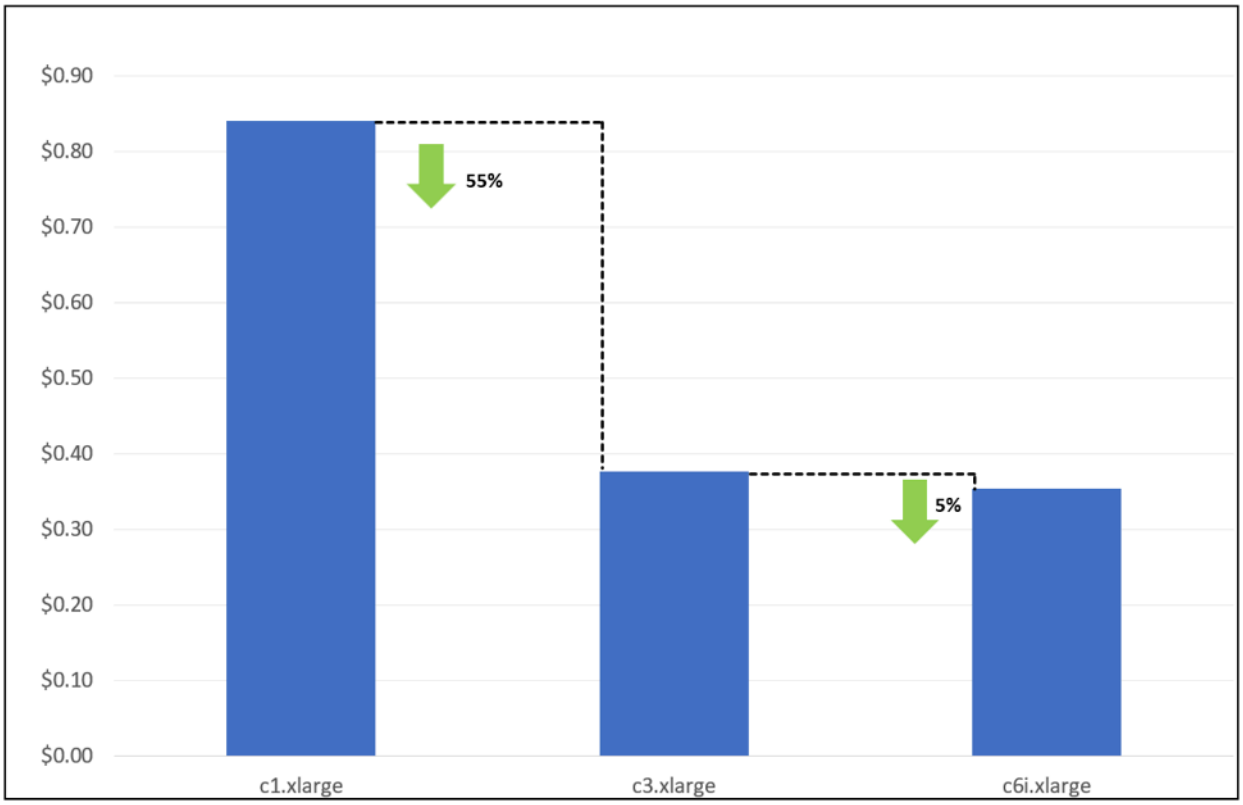
EC2 世代間の価格パフォーマンスの違いを理解する

Amazon の最も一貫した特徴の 1 つは、各新世代EC2が前世代よりも優れた価格パフォーマンスを提供することです。次の表に示すように、新しい世代のEC2インスタンスの料金は、後続のリリースごとに減少します。

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	M1.xlarge	$0.35	r1.xlarge	該当なし
C3.xlarge	$0.21	M3.xlarge	$0.266	r3.xlarge	$0.333
C5.xlarge	$0.17	M5.xlarge	$0.192	r5.xlarge	$0.252

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

次の表は、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	M5.xlarge	$0.192	r5.xlarge	$0.252
C6i .xlarge	$0.17	M6i .xlarge	$0.192	r6i.xlarge	$0.252

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

同じコストにもかかわらず、新しい世代では、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 の向上により、優れた価格パフォーマンスを提供しますIOPS。

価格パフォーマンス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、以前の世代よりも最大 55% 高い価格パフォーマンスを提供します。次の表に示すように、x2iedn はすべてのパフォーマンス面 (すべて前世代と同じ料金) の改善を示しています。

インスタンス	時間料金	vCPUs	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 ドル	8	244	2.3 GHz	237GB SSD	10 Gbps	125 MB/秒	7400
x1iedn.2xlarge	1.66 ドル	8	256	3.5 GHz	240GB 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時間単位料金に基づいています。

シナリオ例

配信車両を追跡し、SQLサーバー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を考えてみましょう。がこの会社のパフォーマンスのボトルネックMACOSMEを確認すると、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 インスタンスの強化により、バッファプール拡張機能を使用してSQLサーバーのパフォーマンスと最適化が向上します。これにより、企業は SQL Server Enterprise Edition から SQL Server Standard Edition にダウングレードできます。また、サーバーSQLライセンスを 8 から 4 vCPUs に減らすこともできますvCPUs。

最適化前：

[サーバー]	EC2 インスタンス	SQL サーバーエディション	月額コスト
製品DB1	x1e.2xlarge	エンタープライズ	$3,918.64

[サーバー]	EC2 インスタンス	SQL サーバーエディション	月額コスト
製品DB2	x1e.2xlarge	エンタープライズ	$3,918.64
合計			$7,837.28

最適化後：

[サーバー]	EC2 インスタンス	SQL サーバーエディション	月額コスト
製品DB1	x2iedn.xlarge	標準	$1,215.00
製品DB2	x2iedn.xlarge	標準	$1,215.00
合計			$2,430.00

これらを合計すると、x1e.2xlarge インスタンスから x2iedn.xlarge インスタンスに変更することで、このシナリオ例では、本番稼働用データベースサーバーで月額 5,407 ドルを節約できます。これにより、ワークロードの総コストが 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時間単位料金に基づいています。

新しいインスタンスへの移行

古い世代の Amazon は Xen ハイパーバイザーでEC2実行され、新しい世代は AWS Nitro System で実行されます。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配信します。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 AMIs 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ています。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。

2018 年 8 月以前に作成されたカスタム Windows AMIsまたは Amazon AMIs が提供する Windows からインスタンスを起動する場合は、Amazon EC2ドキュメントの最新バージョンのインスタンスタイプへの移行の手順を完了することをお勧めします。

バーストインスタンスを使用する

バースト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。

- Desktop Experience を使用する Windows Server の最小仕様には、2 GB の が必要です RAM。t3.micro インスタンスや t3.nano インスタンスは、Windows Server で使用しないようにします。最小量の がないからですRAM。
- 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ほどアイドル状態にならない場合、通常のEC2インスタンスを使用すると、バースト可能なインスタンスを使用するよりも効率的です。クレジットをモニタリングCPUして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、サーバーでバーストインスタンスを使用しないことをお勧めします。SQL サーバーのライセンスは、インスタンス vCPUs に割り当てられた の数に基づいています。SQL サーバーが 1 日の大半アイドル状態の場合、完全に利用されていない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サーバー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。

次のステップ

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。

- 最新世代のEC2インスタンスを使用して、最適な価格のパフォーマンスを実現します。
- 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。インスタンスストレージボリュームなど、SQLサーバーワークロードを強化するための追加オプションや、はるかに大きなvCPU-to-RAM比率のEC2インスタンスを利用できます。ワークロードを徹底的にテストし、 などのモニタリングツールを使用して必要な調整 AWS Compute Optimizer を行うことをお勧めします。

ワークロード	代表的な例	オプションです。
アクティブディレクトリ	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 サーバー	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、通常、プロセスには単純なサーバー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「インスタンスタイプを変更する」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、以下を検討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EBS前に、Amazon でバックアップされている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は、必ずダウンタイムに備えてください。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「インスタンスを停止して起動する」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 リリースし、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。
- インスタンスで休止が有効になっている場合、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 は停止したインスタンスを異常としてマークし、終了して代替インスタンスを起動することがあります。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 Auto Scaling ドキュメント」の「Auto Scaling グループのプロセスを停止して再開Auto Scaling」を参照してください。 EC2
- NVMe インスタンスストアボリュームを持つインスタンスのインスタンスタイプを変更すると、Amazon Machine Image (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使用可能になるため、更新され

たインスタンスにインスタンスストアボリュームが追加され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。

追加リソース

- Amazon EC2インスタンスタイプ (AWS ドキュメント)
- AWS 最適化とライセンス評価 (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の持ち込み

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、これらのワークロードをサポートするいくつかの AWS オプションから選択できます。これには、含まれるライセンス (によって提供される AWS) や、Bring Your Own License (BYOL) オプションが含まれます。Amazon EC2 Dedicated Hosts を使用して、既存の Microsoft ライセンス契約を最大限に活用し、Windows Server を に持ち込むことができます AWS。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 に持ち込むと、大幅なコスト削減につながります。Windows Server をデフォルトの (共有) テナンシー に持ち込むことができないため、 で 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。

Dedicated Hosts は Windows Server BYOLインスタンスだけのものではありません。また、既存のSQLサーバー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。Dedicated Hosts 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サーバーをライセンスできるようにします。これは、SQLサーバーライセンスがインスタンスCPUsに割り当てられた仮想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サーバーワークロードのライセンスを付与できます。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コスト削減に加えて、デフォルトの (共有) テナンシーと比較してSQLサーバー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「サーバー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 Dedicated Hosts

Amazon EC2 Dedicated Host は、EC2コンピューティングサービスの実行に AWS を使用するホストと基本的に同じEC2です。違いは、これらのホストは 1 人の顧客に完全に専念しており、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していることです。Dedicated Hosts を使用すると、リソースを他の AWS 顧客と共有する代わりに、使用に専念している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細かく制御でき、Windows Server や Server などの独自のソフトウェアライセンスを SQLに持ち込むことでコストを削減できます AWS。

以下に留意してください。

- Dedicated Host は、1 人の顧客に完全に専念した物理サーバーです。Dedicated Host のソケットと物理コアを可視化できるため、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 Dedicated Hosts は、異種 Dedicated Hosts 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、および 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 Dedicated Host で 1 つのインスタンスサイズのみをサポートします。これらは同種専有ホストと呼ばれます。
- Dedicated Hosts には、ホストごとに課金されます。つまり、専有ホスト上で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。Dedicated Host の料金は、選択したインスタンスファミリー、リージョン、および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実現できます。

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 Dedicated Host 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 つのソケット、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 Dedicated Host で起動できるインスタンスの数は、インスタンスのサイズによって異なります。

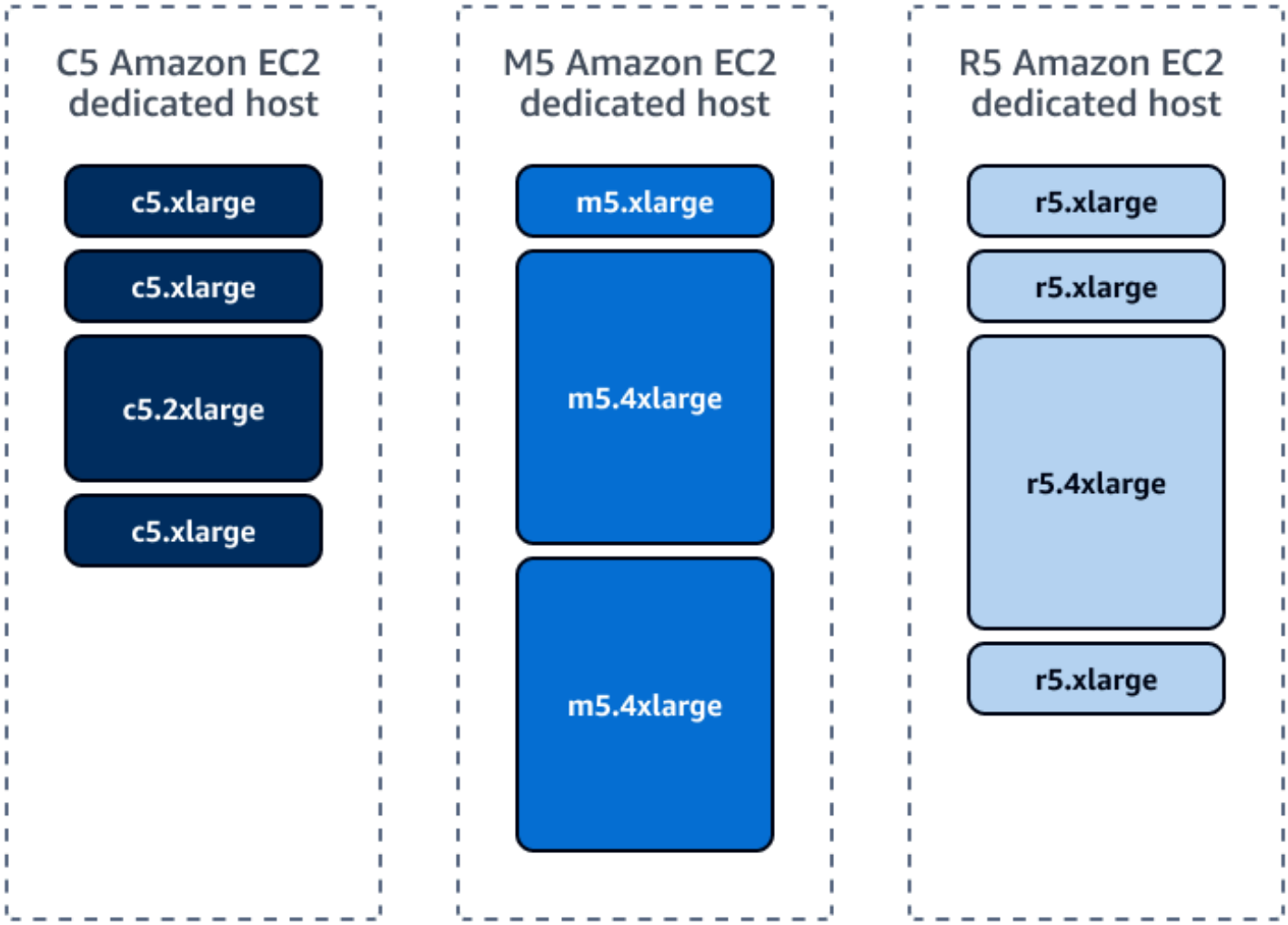
例:

- xlarge (4 vCPUs) の場合、この Dedicated Host で最大 32 m6i.xlarge インスタンスを起動できます。
- 8xlarge (32 vCPUs) の場合、この Dedicated Host で最大 4 m6i.8xlarge インスタンスを起動できます。
- メタル (128 vCPUs) の場合は、この Dedicated Host で最大 1 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 Dedicated Host 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 Dedicated Hosts は、異種 Amazon EC2 Dedicated Hosts と呼ばれます。次の図は、2xlarge、xlarge、4xlarge など、さまざまなインスタンスサイズを持つ C5, M55、R5 専有ホストの例を示しています。



専有ホスト管理

Amazon EC2 Dedicated Hosts 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。

- Dedicated Hosts を最大限に活用するには、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削減につながります。ビジネスユニット間で Dedicated Host を共有することで、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。の組織の一部であり AWS Organizations 、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 を使用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 Dedicated Hosts で Windows Server 2022 を実行できますBYOL。Dedicated Hosts で Windows

Server 2022 を使用する場合は、Windows Server 2022 ライセンスに含まれる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、一意のライセンス条件をエミュレートするための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。これにより、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改善されます。
- 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、管理のオーバーヘッドが軽減されます。License Manager 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、組織全体の増分ライセンス購入、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 で一元管理できます AWS アカウント。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 つのリージョン内の専有ホストの高可用性を設計する場合は、本番稼働に不可欠なワークロード用に、少なくとも 2 つのアベイラビリティーゾーンに少なくとも 2 つの専有ホストを割り当て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 Reference deployment」を参照してください。
- Dedicated Host インスタンスファミリーごとに、インスタンスサイズごとに実行できるインスタンスの数に制限があります。詳細については、Amazon EC2ドキュメントの「Dedicated Hosts Configuration Table」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、次のプライマリ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、使用分のみに料金を支払い、オンデマンドでライセンスを購入して使用できます。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。さまざまな Windows Server、SQLServer、およびその他の Microsoft 製品から選択できます。
- BYOL ライセンスモビリティを備えた製品 – 既存のライセンスがあり、クラウドで使用したい場合は、Microsoft License Mobility 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付きSQLサーバー (SA) などの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を削減できます。

- BYOL ライセンスモビリティのない製品 – 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、Dedicated Hosts 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。Dedicated Hosts は、ほとんどの場合実行される安定した予測可能なワークロードに最適です。

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、ライセンス最適化の最も効果的な戦略の 1 つです。既存の投資を活用し、 AWS コストを削減できるためです。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 日より前に永続的ライセンスを購入しているか、2019 年 10 月 1 日より前に有効なエンタープライズ登録で補正発注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なバージョンにのみライセンスをアップグレードできます。例えば、2017 年に SA を削除した場合、2019 年ではなく、Windows Server 2016 までのみデプロイする権限があります。ただし、2019 年は BYOLから 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、 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 AWS。独自のライセンスをお持ちになる場合、クラウドで実行されているインスタンスに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。

次の表は、さまざまな設定で 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。

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

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

既存のライセンスを使用して、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 Dedicated Hosts BYOLで 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参する必要があります。BYOL では、 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取得するには、ソフトウェアベンダーからのライセンス権限があり、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、Dedicated Hosts でインスタンスを起動できます。の作成の詳細についてはBYOLAMI、 AWS ブログの Microsoft Workloads の「VM Import/Export を使用してオンプレミスAMIsから Windows Server Bring-Your-Own-License を作成する方法」を参照してください。

> **Note**
>
> Auto に設定されたライセンスタイプは、AWS ライセンスに含まれるオプション 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ジングと最適化は、 のコスト最適化の重要な要素です AWS。適切な戦略を実装すると、Amazon EC2 Dedicated Hosts と BYOLオプションを使用して、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大限の価値を実現できます。

このセクションでは、次のシナリオ例について説明します。

- T3 Dedicated Hosts によるコスト削減
- SQL サーバーとの共有テナンシーと専有ホストの比較 BYOL
- 高可用性SQLサーバーデプロイ

T3 Dedicated Hosts によるコスト削減

T3 Dedicated Hosts は、従来固定CPUリソースを提供する他の Amazon EC2 Dedicated Hosts とは異なります。対照的に、T3 Dedicated Hosts は、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できるバースト可能なインスタンスをサポートします。オーバーサブスクリプションとも呼ばれるCPUリソースの共有は、単一の T3 Dedicated Host が同等の汎用 Dedicated Host の最大 4 倍のインスタンスをサポートできるようにするものです。

T3 Dedicated Hosts は、他の Amazon EC2 Dedicated Host よりも高いインスタンス密度を提供することTCOで、 を低くします。バースト 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平均CPU使用率で low-to-moderate、より多くのインスタンスを統合することができます。T3 Dedicated Hosts は、他の Amazon EC2 Dedicated Hosts よりも多くの vCPU とメモリの組み合わせで、より小さなインスタンスサイズも提供します。インスタンスサイズが小さいTCOと、オンプレミスホストと同等以上の統合率が低くなり、実現に役立ちます。

T3 Dedicated Hosts は、Microsoft Windows デスクトップ、Windows Server、サーバー、SQLOracle Databases など、ソケットごと、コアごと、または VM ごとのBYOLソフトウェアライセンスが利用可能な、 low-to-moderateCPU使用率の高いソフトウェアの実行に最適です。

T3 Dedicated Hosts 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CPUsで物理を簡単にサブスクライブし、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 RAMVMwareホストを使用しています。さらに、各ホストは 96x2 v 、4 GB の RAM Windows Server 仮想マシンを実行しておりCPU、平均CPU使用率は低くなります。

仮想マシンを T3 Dedicated Hosts 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 Dedicated Hosts は、現在のオンプレミスVMwareホストRAMの 2 倍の量です。T3 Dedicated Hosts 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 Dedicated Hosts の使用によるコスト削減を示しています。

	オンプレミス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VMware ホスト	T3 専有ホスト	削減量
ホストあたりの物理コア数	36	48	
RAM ホストあたり (GB)	384	768	
2 v CPU、ホストRA MVMsあたり 4 GB	96	192	
の合計数 VMs	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 サーバーとの共有テナンシーと専有ホストの比較 BYOL

Amazon EC2 Dedicated Hosts の価値を示すための実用的な例を考えてみましょう。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サーバーワークロードを実行し、同じワークロードを にコスト効率良くデプロイしたいと考えています AWS。この組織が独自のライセンス (BYOL) をお持ちの場合、SA に対して引き続き支払いを行い、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 のエンタイトルメントと 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を適正にサイズ設定することで、SQLサーバーライセンスを 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 ドルになります。

Amazon EC2 Dedicated Hosts を使用すると、物理コアレベルでSQLサーバーをライセンスできます。これは、SQLサーバーライセンスがインスタンス vCPUs に割り当てられた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と、共有テナンシーで vCPUs 必要な 140 コアではなく、96 コアをカバーするだけで済みます。R5 Dedicated Hosts をデプロイし、物理レベルでワークロードをライセンスすることで、必要な数の SQL Server Enterprise Edition ライセンスを 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、SQLサーバーワークロードを最大 192 コア (ハイパースレッディングに対応) までデプロイできます。

この場合、組織は年間約 341,000 ドルの SA コストを支払います。共有テナンシーで適切なサイジングを行った後、コストを 140 で 197,000 USD に削減しますvCPUs。Amazon EC2 Dedicated Hosts はさらにコストを 151,000 USD に削減します (約 56% の減少) 。

高可用性SQLサーバーデプロイ

この例では、コストが でのSQL AWS サーバーデプロイにどのように影響するかを、ライセンスに関するさまざまな考慮事項とともに分析します。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、RAMそれぞれ 16 GB vCPUs と 256 GB があります。以下のシナリオの詳細を参照してください。

- サーバー – SQLサーバー
- オペレーティングシステムエディション – Windows Server Datacenter 2019
- SQL Server Edition – 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 することなく 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正に調整することをお勧めします。ワークロードのサイズを適正に調整したら、16 の を提供する x2iedn.4xlarge インスタンスタイプに配置しますvCPUs。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の 2 倍も含まれます。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 の両方に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 Server Enterprise SQL サーバーをデプロイします。このオプションでは、Windows

ライセンスと 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。以下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
- 単位あたりの月額コスト (USD) – $7,351.47
- サーバー数 – 6
- CPU – 16
- メモリ – 512
- サーバー 6 台の月額料金 – $44,108

シナリオ 2

組織には、共有テナンシーに SA と BYOL for SQL Server 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vCPUsに割り当てられた数に基づいて独自のSQLサーバー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、 vCPUs それぞれが 16 であるため、合計 96 vCPUs が必要です。以下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
- 単位あたりの月額コスト (USD) – $2971.47
- サーバー数 – 6
- CPU – 16
- メモリ – 512
- BYOL コア – 96
- サーバー 6 台の月額料金 – 17,828 USD

SA に独自のSQLサーバーライセンスを導入することで、このシナリオの組織はSQL、サーバーにライセンス込みオプションを使用するよりも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、SQLServer 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 Dedicated Hosts BYOL 上の Windows と SQL Server の両方の 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 Dedicated Hosts を使用します。各ホストには 24 個の物理コアと 48 個の がありますvCPUs。これにより、RAMそれぞれ 16 GB vCPUs と 256 GB を持つ 6 つの SQL Server Enterprise サーバーに十分な容量が提供されます。以下のシナリオの詳細を参照してください。

- 専用ホストの数 – 2
- インスタンスファミリー – x2iezn
- 時間単位のコスト (USD) – $11.009
- 単位あたりの月額コスト (USD) – $8,036
- 物理コア – 48
- 使用可能な vCPU – 96
- Windows Server コアライセンスが必要 – 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 ファミリーの Amazon EC2 Dedicated Hosts 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 Pricing Calculator 見積りを参照してください。このシナリオの組織は、Windows ライセンスを導入することで、月額 1,755.65 ドルを節約できます。Amazon EC2 Dedicated Hosts を使用している場合は、必要なSQLサーバー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、6 つの SQL Server Enterprise サーバーを vCPUs 16 個ずつカバーするには、96 個の SQL Server Enterprise ライセンスが必要です。ただし、Amazon EC2 Dedicated Hosts と物理コアレベルでのライセンスを使用することで、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。

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

- オンプレミスサーバー – SQLサーバー
- vCPU – 16
- メモリ – 256
- サーバー数 – 6
- シナリオ 1 の月額コスト: Windows (LI) + SQL Server Enterprise (LI) – $44,108
- シナリオ 2 の月額コスト: Windows (LI) + SQL Server Enterprise (BYOL ） – $17,828
- シナリオ 3 の月額コスト: Amazon EC2 Dedicated Host での Windows (LI) + SQL Server Enterprise (BYOL ） – $16,073

> **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 Dedicated Reserved Instances 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 年の任期をコミットできます。詳細については、このガイドの「Amazon での Windows の支出の最適化EC2」セクションを参照してください。

Amazon EC2 Dedicated Hosts 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)

が Dedicated Host の料金をサポートするAWS Pricing Calculatorようになりました。これにより、適切な基盤となる Dedicated Host を選択できます。

コスト最適化に関する推奨事項

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 Cost Explorer。

1. Cost Explorer を有効にする。
2. Cost Explorer を使用して、Amazon EC2 Dedicated Host 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. を実行していることを検証しますBYOL。Amazon EC2コンソールのインスタンスまたはAMIページ、または `describe-images` または `describe-instances` コマンドによって返されるレスポンスに、次のプラットフォームの詳細と使用状況オペレーション値を表示できます。
   - プラットフォームの詳細: Windows 、使用状況オペレーション： RunInstances:0002 (ライセンス込み)
   - プラットフォームの詳細： Windows BYOL、使用状況オペレーション： RunInstances:0800

追加リソース

- ライセンスタイプ変換の対象となるライセンスタイプ (AWS License Manager ドキュメント)
- AWS License Manager & Dedicated Host 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)
- VM Import/Export を使用してオンプレミスAMIsから Windows Server Bring-Your-Own-License を作成する方法 (ブログの AWS Microsoft Workloads)
- VM のインポート/エクスポート (AWS ドキュメント)
- Amazon Web Services 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)
- Amazon EC2 Dedicated Hosts に高可用性SQLサーバーをデプロイする (AWS クラウドオペレーションと移行ブログ)

Amazon での Windows の支出を最適化する EC2

概要

サーバーを に移行することに関する最大の懸念の 1 つは、インフラストラクチャコスト AWS です。クラウドの利点の 1 つがオンデマンドでリソースに料金を支払っていることは事実ですが、24 時間 365 日利用可能な本稼働ワークロードがあります。Savings Plans は、EC2インスタンス間での定常状態 AWS の使用量 AWS Lambda、および の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EC2、一貫した SageMaker 使用量 (1 時間あたり 10 ドルなど) へのコミットメントと引き換えに、Amazon 、Fargate、Lambda、Amazon の使用量の料金を下げるのに役立ちます。1 年または 3 年にわたって一貫した時間単位のコンピューティング支出を約束し、その代わりにその使用量の割引を受け取ります。

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。

- No Upfront オプション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は、 Savings Plans の料金が安くなります。コミットメントの半分以上が前払いされ、残りの料金は毎月請求されます。
- All Upfront オプションは最低料金で、コミットメント全体が 1 回の支払いで請求されます。

Savings Plans の有効期限と今後のキューに入れられた Savings Plans は、 で追跡できます AWS Cost Explorer。Savings Plans アラートを使用すると、プランの有効期限の 1、7、30、60 日前、またはコミットメントが購入キューに入れられたときに、事前 E メールアラートを受信できます。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、オンデマンドレートと Savings Plans レートがあります。コンピューティング使用量を 1 時間あたり 10 USD にコミットすると、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 料金が発生します。コンピューティング支出コミットメントを超える使用量は、通常のオンデマンド料金で課金されます。Savings Plans の使用を開始するには、 の Cost Explorer を使用します AWS Management Console。

Cost Explorer で提供されている推奨事項を使用して最大の節約を実現することで、Savings Plans に簡単にコミットできます。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、次に統合請求ファミリー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、支払人アカウントの 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。これらのレコメンデーションは、最適な Savings Plans を簡単に購入できるように、いつでも更新できます。

Compute Savings Plans

Compute Savings Plans は、最も柔軟性が高く、コスト削減に役立ちます。これらのプランは、EC2 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、テナンシーに関係なく、インスタンスの使用に自動的に適用されます。また、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 インスタンスに変更したり、ワークロードを EU (アイルランド) から EU (ロンドン) に移動したりEC2、ワークロードを Fargate または Lambda に移動したりできます。Savings Plans の料金は、引き続き自動的にお支払いいただきます。

EC2 インスタンス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に対するコミットメント (バージニア北部で一貫したレベルの M5 使用量をコミットするなど) と引き換えに、最も深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に対する割引が自動的に提供されます。EC2 Instance Savings Plans を使用すると、そのリージョンのファミリー内のインスタンス間の使用状況を変更できます。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 に移行し、Savings Plans の料金から自動的にメリットを得ることができます。

Compute と EC2 Instance Savings Plans はどちらも、Amazon 、Amazon Elastic Kubernetes Service (Amazon EKS) EMR、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 のECS料金は Savings Plans の対象外ですが、基盤となるEC2インスタンスは です。EC2 Compute Savings Plans は適用性が広いため、インスタンス Savings Plans は Compute Savings Plans の前に適用されます。 Compute Savings Plans

> **Note**
>
> Savings Plan sは、コミットメントの実行後に簡単に変更することはできません。Savings Plans オプションのいずれかにコミットする前に、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料金よりも低価格を提供し、期間中にキャンセルすることはできません。

時間単位のコミットメントの例

Savings Plan を購入する場合、プランの有効期間に対して 1 時間ごとに金銭的なコミットメントを行います。コンピューティング使用量を 1 時間あたり 10 USD にコミットすると、Savings Plan の料金は、1 時間あたり最大 10 USD のすべての使用量に自動的に適用されます。コミットメントを超える使用量は、通常のオンデマンド料金で課金されます。Cost Explorer の Savings Plans 購入レコメンデーションツールを使用して、節約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財務コミットメントは、プランの期間中は変更できません。使用量を分析した後にコミットメントを増やす場合は、余分な使用量をカバーする追加の Savings Plan sを購入できます。

Savings Plans の利点

Savings Plans は、リザーブドインスタンスと比較して、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 – 金銭的コミットメントと引き換えに、自動割引を受け取ります。
- 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。
- 節約の可能性 – 節約する方法はさまざまです。次の例を考えます。
  - Compute Savings Plans (d2.8xlarge、3 年、すべて前払い、Windows、共有テナンシー、us-east-2) を使用した Windows Server ワークロードの 60% 削減
  - EC2 Instance Savings Plans (d2.8xlarge、3 年、すべて前払い、Windows、共有テナンシー、us-east-2) を使用した Windows Server ワークロードの 73% 削減
  - エキゾチックではないインスタンスタイプ (t3 ファミリー、3 年、すべての前払い、ウィンドウ、共有テナンシー、us-east-2) が 28～41% 削減
  - Windows Server の平均 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。割引価格での使用をコミットします。

すべてのタイプのコンピューティング使用量には、Savings Plan レートとオンデマンドレートがあります。次の表は、すべてのオペレーティングシステムタイプの Savings Plans とオンデマンド料金を示しています。Savings Plans 料金はコミットされた使用量に対して課金され、コミットメントを超えた使用量は通常のオンデマンド料金で課金されます。

Instance name	Savings Plans レート	オンデマンドの節約	オンデマンドレート	オペレーティングシステム	リージョン	お支払い方法	期間の長さ
x2iedn.xlarge	$0.32	61%	$0.83	Linux	米国東部 (バージニア北部)	前払いなし	3
x2iedn.xlarge	$2.01	50%	$1.02	Windows	米国東部 (バージニア北部)	前払いなし	3
x2iedn.xlarge	$1.02	20%	2.52 ドル	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	61%	$0.83	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、 には別の割引がありますBYOL。これらはすべてCompute Savings Plans 計算ツール で分類されます。

リザーブドインスタンスの料金モデル

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になる可能性があります。Savings Plans は、標準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 は、EC2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、またはリージョンに関係なく、インスタンス使用量に対して低価格を提供します。また、最大限の柔軟性も実現します。

次の表は、Savings Plans とリザーブドインスタンスのどちらを選択するかに役立ちます。

	Reserved Instance	EC2 インスタンスSavings Plans	Compute Savings Plans
平均 1 年割引	最大 38%	最大 29%	最大 29%
平均 3 年割引	最大 58%	最大 73%	最大 60%
インスタンスファミリー	Fixed	Fixed	柔軟性
インスタンスサイズ	固定 (Linux 以外)	柔軟性	柔軟性
地域別	1 リージョン	1 リージョン	柔軟性
オペレーティングシステム	Fixed	柔軟性	柔軟性
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、一部、前払いなし	すべて、一部、前払いなし	すべて、一部、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、時間単位の金銭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。時間単位の財務コミットメントは、プランの有効期間中にキャンセルまたは変更することはできませんが、追加の使用量をカバーする追加の 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貫した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 Intelligence Dashboards 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ラインを提供します。ワークロードの 75% が定常状態の場合、75% が適切なターゲットです。これにより、動的なワークロードに基づくオンデマンド/可変支出の 25% が残ります。これを 85% まで引き上げる必要がある場合は、Savings Plans のコミットメントをもう 1 つ購入して、時間単位の金銭的コミットメントを増やすことができます。

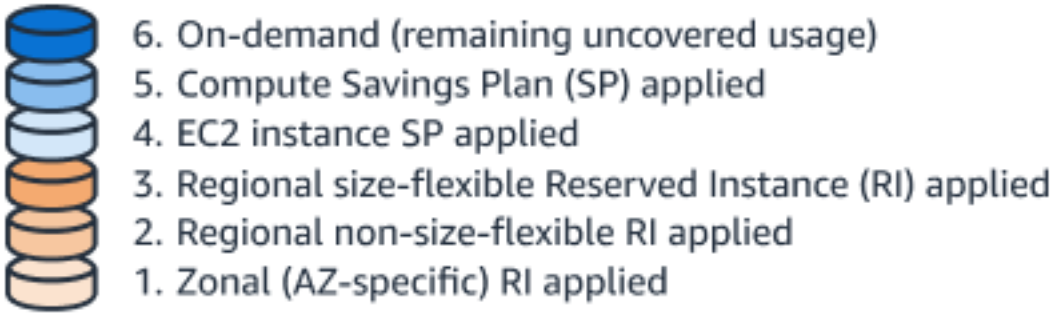
> **Note**
>
> リザーブドインスタンスの代わりに Savings Plans を購入することをお勧めしますが、リザーブドインスタンスを既に購入している場合は、2 つのコミットメントモデルを連携させることができます。

リザーブドインスタンスを購入したが、Savings Plans オプションを試し始める例を考えてみましょう。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。以下は、 に適用できる階層です AWS アカウント。

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の残り時間が 時間の場合、組織の残りの部分に適用されます。
2. Windows 用のサイズ変更不可能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の一致する使用状況に適用されます。残りのものは、組織の残りの部分にロールアウトされます。
3. サイズを柔軟に調整できる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 (ファミリー内の最小インスタンスからより大きなインスタンスまで) に適用され、その後、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、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、割引率が最も高くなる使用量から始まり、次に割引率が最も低くなります。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, M6C5 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 から除算した後の料金を示しています。Compute と EC2 Instance Savings Plans の両方が、最初に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 Dedicated Hosts 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 Dedicated Hosts

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

- 16 vCPU と 128 GB の 2 つのサーバー RAM
- 32 vCPU と 164 GB の 2 つのサーバー RAM
- 8 vCPU と 64 GB のサーバー 1 台 RAM
- vCPU と 32 GB を持つ 16 台のサーバー RAM

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

インスタンスタイプ	CPU	RAM	量
r5.4xlarge	16	128	2
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 Dedicated Hosts に分散できることを示しています。次の表は、これら 2 つの Dedicated Host のコストを示しています。

Dedicated Host オンデマンドシナリオ	前払い	1 か月	1 年	3 年	AWS Pricing Calculator
オンデマンド	なし	10,123 ドル	121,475 ドル	364,392 ドル	AWS Pricing Calculator 見積り
1 年間の Savings Plan	なし	7,447 ドル	89,362 ドル	–	AWS Pricing Calculator 見積り
3 年間の Savings Plan	なし	5,476 ドル	65,712 ドル	197,128 ドル	AWS Pricing Calculator 見積り
前払いによる 3 年間の Savings Plan	84,438 ドル	2,755 ドル	117,499 ドル	183,618 ドル	AWS Pricing Calculator 見積り

に移行するサーバーがある場合 AWS、1 年間の Savings Plan の最終料金は、オンデマンド料金の 121,475 ドルではなく、89,362 ドルです。これは、1 年後の 26.5% の割引を表します。 AWS より長期間の滞在を検討している場合は、3 年間のSavings Planを選択して、さらに深いコスト削減を実現できます。3 年後には、364,392 ドルではなく 197,128 ドルを支払います。これにより、3 年後に総額の 46% を節約できます。

ライセンスを含む 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 GB vCPUs と 4 GB の 2 つのウェブサーバー RAM
- 8 GB vCPUs と 16 GB の 2 つのアプリケーションサーバー RAM
- 16 GB vCPUs と 64 GB の 2 つのデータベースサーバー RAM (SQLServer Standard Edition を使用)

次の表は、 で使用できるサーバーインスタンスを示しています AWS。

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

次の表は、 内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 Pricing Calculator
オンデマンド	なし	3,912 ドル	46,950 ドル	140,849 ドル	AWS Pricing Calculator 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 Pricing Calculator
1 年間の Savings Plan	なし	3,466 ドル	41,952 ドル		AWS Pricing Calculator 見積り
前払いなしの 3 年間の Savings Plan	なし	3,189 ドル	38,264 ドル	114,804 ドル	AWS Pricing Calculator 見積り
前払いによる 3 年間の Savings Plan	112,110 ドル	なし	なし	なし	AWS Pricing Calculator 見積り

これらのサーバーをオンデマンド料金で本番稼働環境 (24 時間年中無休) で実行する場合は、月額 3,912 ドルをお支払いいただきます。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 Plan を選択した場合、月額料金は 3,466 USD に減少します。初年度の終了時には、41,952 ドルを支払います。これは合計 11% の割引です。前払いなしで 3 年間の Savings Plan sを選択した場合、月額コストは 3,189 ドルに削減されます。3 年後には、114,804 ドルを支払います。これにより、18.5% の節約になります。

コスト最適化に関する推奨事項

どちらのシナリオも、 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 AWS。2 番目のシナリオの割引は、1 番目のシナリオに比べ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。ライセンス料金には割引 AWS は適用されませんが、いつでも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。

- アクセスレコメンデーション
- ニーズに合わせてレコメンデーション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

アクセスレコメンデーション

Amazon EC2コンソールを使用して、Savings Plan のレコメンデーションにアクセスできます。レコメンデーションをダウンロードして、後で CSV形式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「Savings Plans のモニタリング」を参照してください。Savings Plans

ニーズに合わせてレコメンデーション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、Savings Plansを選択します。このページでは、レコメンデーションを行う前後のインスタンスとコンピューティング料金を示します。レコメンデーションには、次の要素を調整することもできます。

- 用語 – 例えば、1 ~ 3 年
- 支払いオプション – 例えば、前払い 、一部前払い 、前払いなし
- 履歴 – 過去 7、30、60 日など

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 日実行されているインスタンスがあるとします。Savings Plan を使用することをお勧めします。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 ドル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、定義したしきい値を下回る場合は、使用率と力バレッジを追跡し、予算が終了したときにアラートを設定することもできます。

レコメンデーションを確認する

Savings Plan のレコメンデーションを慎重に確認することをお勧めします。 AWS は許可なく変更することはありません。これらはレコメンデーションのみであり、適用するかどうかはユーザー次第です。

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、Savings Plansを選択します。次に、「Purchase Savings Plans」を選択します。必要に応じて、用語、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 、EC2Instance Savings Plans 、 SageMaker および 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「 Savings Plans の購入」を参照してください。

使用率レポートを取得する

Savings Plan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、使用率の確認、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の確認、新しい割引のキャンセルまたは追加に役立ちます。このレポートは、 などの他の形式にエクスポートできますCSV。詳細については、「 Savings Plans ドキュメント」の「使用率レポートの使用」を参照してください。 Savings Plans

購入のベストプラクティスに従う

Savings Plans を購入する前に、以下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実行します。
- 30～60 日間一貫して保持する時間単位の料金を設定します。
- 組織が満足しているのと同じくらい一貫した時間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てください。
- Savings Plans 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% 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 Plan sを購入します。

追加リソース

- 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、支出の追跡と報告に役立つ AWS Budgetsなどのツールが含まれます。このセクションでは、支出について AWS Windows 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リソースに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、 を使用して Windows EC2やその他の Microsoft サービスを適切にモニタリングできます AWS Budgets。

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、予想される支出、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下げる場合は、Savings Plan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、継続的にコスト削減を実現できます。 AWS Cost Explorer を使用して Savings Plan s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 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AWS Cost Anomaly Detection。

コスト最適化の推奨事項

、Cost Explorer AWS Budgets、および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、一般的な使用支出ではなく、リソースを分類してグループ化し、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

オートメーション 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。

AWS Budgets、Cost Explorer 、および Cost Anomaly Detection で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。タグを使用して、これらのタグに一致する項目の特定の予算を設定して、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 タグを監視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にアラートを送信 AWS Config し、Windows のEC2支出を正確に表現できます。

タグを配置したら、 でカスタム予算を作成できます AWS Billing。これにより、Windows のEC2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。

を使用してアラートを設定する AWS Budgets

この例では、Windows の日次予算を作成しますEC2。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的な予算です。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 Management Console 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、[Budgets] を選択します。
3. ページの上部で、[Create budget] 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. UnderBudget タイプ で、Cost budget を選択します。[次へ] を選択します。
6. UnderDetails 、予算名 には、予算の名前を入力します。例えば、Windows は をEC2消費します。
7. UnderSet 予算額 、 forPeriod 、Daily を選択します。
8. ForBudget 更新タイプ では、予算期間後にリセットされる予算の繰り返し予算を選択します。
9. 開始日 では、開始日または期間を選択して、予算額に対する追跡を開始します。
10. バジェットメソッド で、自動調整 (新規) を選択します。
11. ベースライン時間範囲 では、カスタム範囲 を選択し、30 日を入力します。
12. [Next (次へ)] を選択します。

13.Budget スコープセクションで、特定の AWS コストディメンションをフィルタリング を選択します。これは、タグを使用して適切なディメンションを作成する場合です。 AWS Budgets は、フィルター内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.フィルターの追加 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.[Next (次へ)] を選択します。

17.アラートの設定ページで、アラートしきい値の追加 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。50% のしきい値アラートが月の中間ポイントより前に違反すると、警告が表示されます。そうすることで、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、Actual を選択します。

20.E メール受信者 には、E メールアドレスを入力します。しきい値 100 に別のアラートを追加します。

> **Note**
>
> この例では、アラートにシンプルな E メール通知を使用しますが、Amazon Chime または Slack 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、異常である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。

以前に作成した 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 Management Console 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. Cost Monitor タブを選択し、Create monitor を選択します。
4. ステップ 1 で、コスト配分タグをモニタータイプとして選択します。

5. コスト配分タグキー で、Windows EC2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 に名前を付けるには、Windows EC2支出 と入力します。
8. [Next (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を作成する」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 には、Windows EC2の支出異常 と入力します。
11. アラート頻度 では、日次概要 を選択します。
12. アラート受信者 には、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 の場合は、10 と入力し、予想速度 を超える割合を選択します。
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

アラートは Windows のEC2支出をモニタリングするのに役立つツールですが、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。このビデオでは、Cost Explorer でEC2コストを分析して削減する方法を説明します。詳細については、AWS 「Supports You | Understanding and Reducing Your EC2 Costs」の動画をご覧ください YouTube。

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 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 で次の正しいフィルターを設定する必要があります。

- プラットフォーム では、Windows (Amazon VPC) を選択します。API オペレーション では、RunInstance:0002 を選択します。これは、ライセンスに含まれる 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には、RunInstance:0002 を RunInstance:0800 に変更します。これは Windows EC2 の請求コードですBYOL。

Cost Explorer のこの可視性により、Windows で消費しているものを正確にコストをすばやくフィルタリングできますEC2。 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 で視覚化

できるレポートを生成 QuickSight 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、AWS 「Supports You - Visualizing Your Cost and Usage Reports」の動画をご覧ください YouTube。

追加リソース

- 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 | Amazon Web Services (YouTube)
- AWS Cost and Usage Report クエリライブラリ (AWS Well-Architected Labs)

SQL サーバー

お客様は、他のクラウドプロバイダーよりも 15 年以上 AWS にわたって Microsoft ワークロードを実行しています。これは主に、 クラウドでの Microsoft アプリケーションの使用経験が最も豊富 AWS で、Windows Server と Microsoft SQL Server に最適なプラットフォームを次の分野で提供しているためです。

- パフォーマンスと信頼性の向上
- セキュリティとアイデンティティのサービスの強化
- 移行サポートの強化
- 最も広範で深い機能
- 総所有コストの削減 (TCO)
- 柔軟なライセンスオプション

AWS は、Active Directory、.、SQLServer、NETSQLWindows Desktop as a Service、サポートされているすべてのバージョンの Windows Server など、Server に依存する Windows アプリケーションを構築および実行するために必要なすべてをサポートします。実績のある専門知識 AWS により、Windows ワークロードのリフトアンドシフト、リファクタリング、モダナイズを簡単に行うことができます。

ガイドのこのセクションでは、以下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ソリューションを選択する
- SQL サーバーライセンスを理解する
- SQL サーバーワークロードに適したEC2インスタンスを選択する
- インスタンスの統合
- SQL Server エディションの比較
- SQL Server Developer Edition を評価する
- Linux でSQLサーバーを評価する
- SQL サーバーバックアップ戦略の最適化
- SQL サーバー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サーバーライセンスを最適化する
- Compute Optimizer を使用してSQLサーバーのサイズを最適化する

- SQL サーバーワークロードの 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ソリューションを選択する

概要

復旧時間目標 () や復旧ポイント目標 () など、ディザスタリカバリ (DR) の目的 も満たしながら、AWS ビジネスニーズを満たす でのSQLサーバー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RPO。RTO以下のソリューションは、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と同時に、SQLサーバーのワークロードのコストを最適化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、高可用性とディザスタリカバリ (HA/DR) solutions for SQL Server databases. An availability group consists of a set of user databases that fail over together. Always On availability groups also provide redundancy at the database level, but don't require shared storage—each replica has its own local storage. You can deploy this feature as an HA/DR ソリューション) を提供します。詳細については、Microsoft ドキュメントの「Always On 可用性グループとは」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 は Windows Server フェイルオーバークラスター (WSFC) FCIsを使用してSQL、サーバーインスタンスレベルで HA を提供します。FCIs データベースをホストするには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 クラスターインスタンス (SQL サーバー)」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 の両方にまたFCIがるSQLサーバー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 AWS リージョン。SIOS DataKeeper は、ローカル Amazon Elastic Block Store (Amazon EBS) ボリュームSANを使用してクラスター化された仮想を作成し、リージョンと 間の非同期レプリケーションを使用してディザスタリカバリを行いながら、HA のアベイラビリティーゾーン間で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「Windows アプリケーションの高可用性保護」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 つの個別の Always On 可用性グループにまたがる特殊なタイプ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 (例: `us-east-1`および) にまたがって配置できます`us-west-1`。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。SQL Server Enterprise Edition は、分散可用性グループをデプロイするために必要です。詳細については、Microsoft ドキュメントの「分散可用性グループ」を参照してください。

- ログ配送 – 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、ログ配送を実装して、複数のリージョンにわたってデータベースを保護することができます。トランザクションとログの配送頻度に応じて、 RPO と を数分RTOで実現できます。詳細については、Microsoft ドキュメントの「ログ配送 (SQL サーバー) について」を参照してください。
- AWS Elastic Disaster Recovery – Elastic Disaster Recovery は、DR AWS 目的で、任意の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サービスとしてのソフトウェア (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サーバー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「Elastic Disaster Recovery とは」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 との間で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のSQLサーバーインスタンスを設定できます。詳細については、 AWS DMS ドキュメントの「 とは AWS Database Migration Service」を参照してください。

SQL Server Always On 可用性グループ

Server SQL Enterprise Edition を高可用性 Always On アベイラビリティーグループ のみに使用する場合は、基本的なアベイラビリティー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サーバーエディション間のコスト差の詳細については、このガイドのSQLサーバーエディションの比較セクションを参照してください。

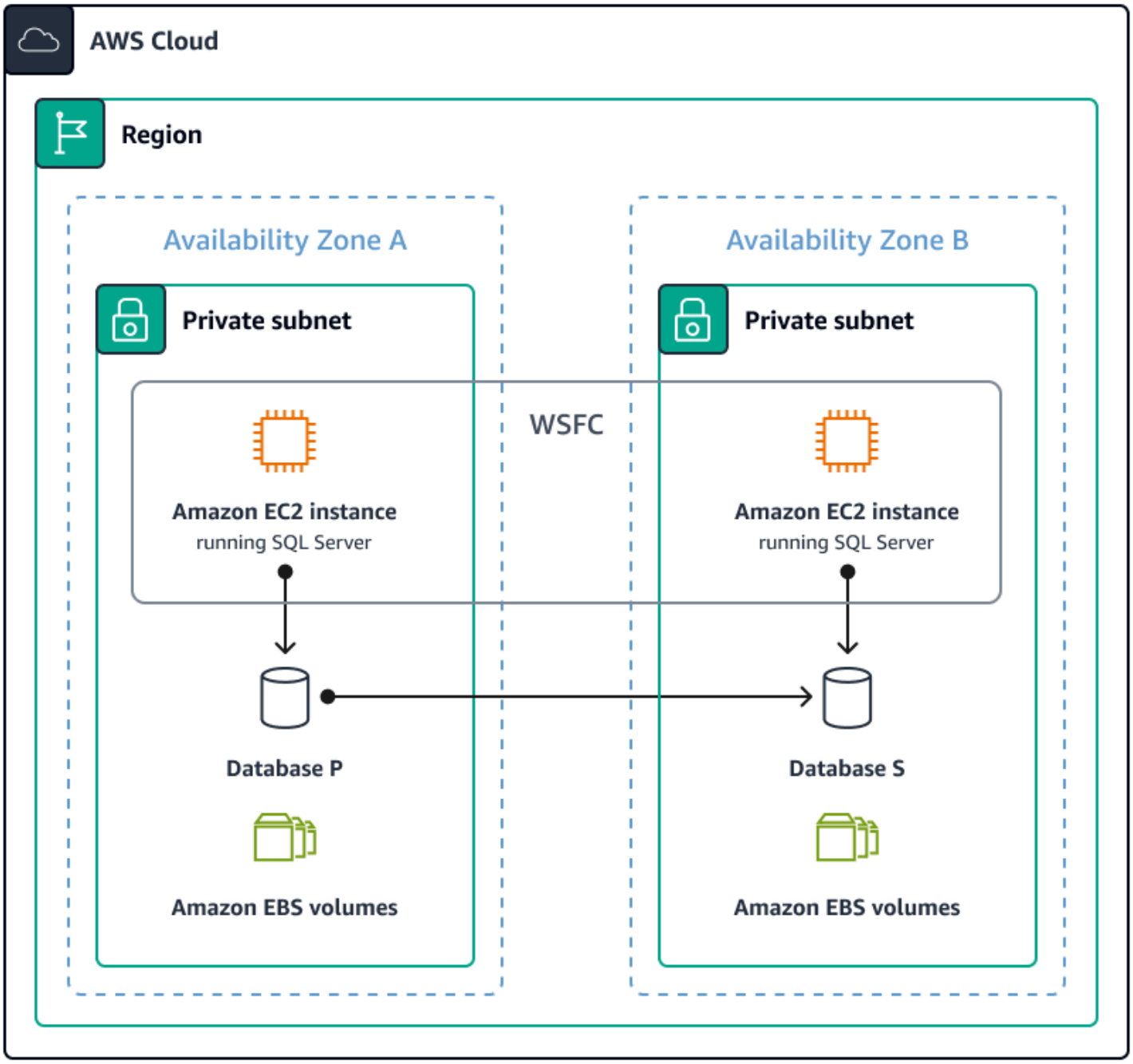
特徴

- SQL Server Standard Edition で利用可能
- 2 つのレプリカの制限 (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

制約事項

- 可用性グループごとに 1 つの可用性データベースのみのサポート
- 基本的な可用性グループは、分散可用性グループの一部にすることはできません

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 (FCIs) 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。FCIs リードレプリカ設定なしで SQL Server データベースの高可用性を求める場合、 は信頼性の高いソリューションを提供します。

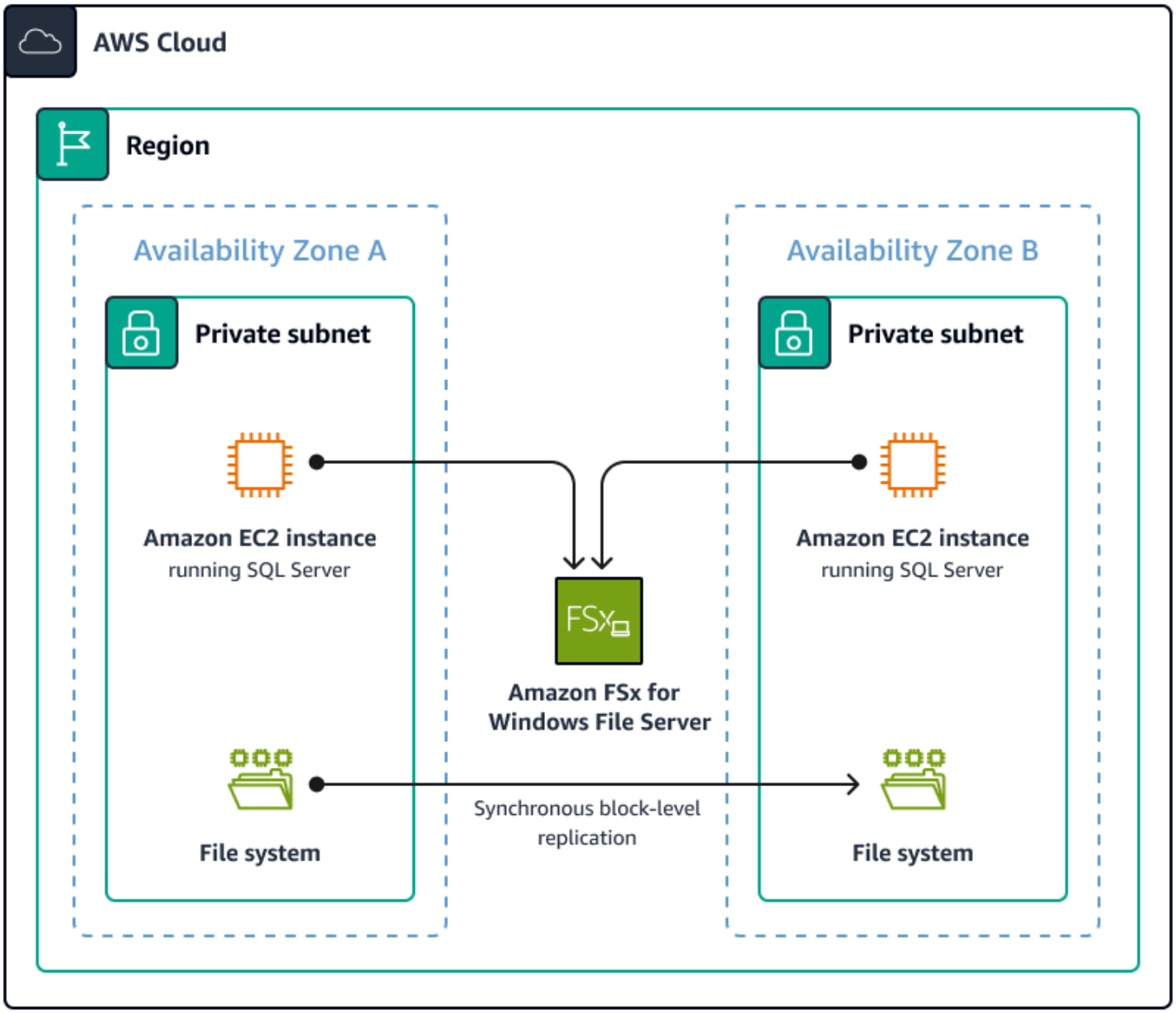
可用性グループとは異なり、 FCIsは SQL Server Enterprise Edition を必要とせずに、信頼性の高いフェイルオーバーソリューションを提供できます。代わりに、SQLサーバースタンダードエディションのライセンスのみFCIsが必要です。を使用して、SQLサーバーのライセンスコストFCIsを 65～75% 削減できます。

> **Note**
>
> SQL Server Editions のコスト差の詳細については、このガイドの「Compare SQL Server Editions」セクションを参照してください。

以下の点を考慮します。

- Amazon FSx for Windows File Server は、SQLサーバー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る可能性があります。詳細については、 AWS ストレージブログの「Amazon FSx for Windows File SQL Server の投稿を使用した Microsoft Server の高可用性デプロイの簡素化」を参照してください。
- Software Assurance のメリットの概要 (ダウンロード可能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SQLライセンスを提供する必要がないため、ライセンスのコスト削減につながります。

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



SIOS DataKeeper

FCIs に SQL Server を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、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) が通常使用されますが、これは では実行可能なオプションではありません AWS。Amazon FSx for Windows File Server は、 FCI上の SQL Server 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できないようにするための制限があります AWS リージョン。

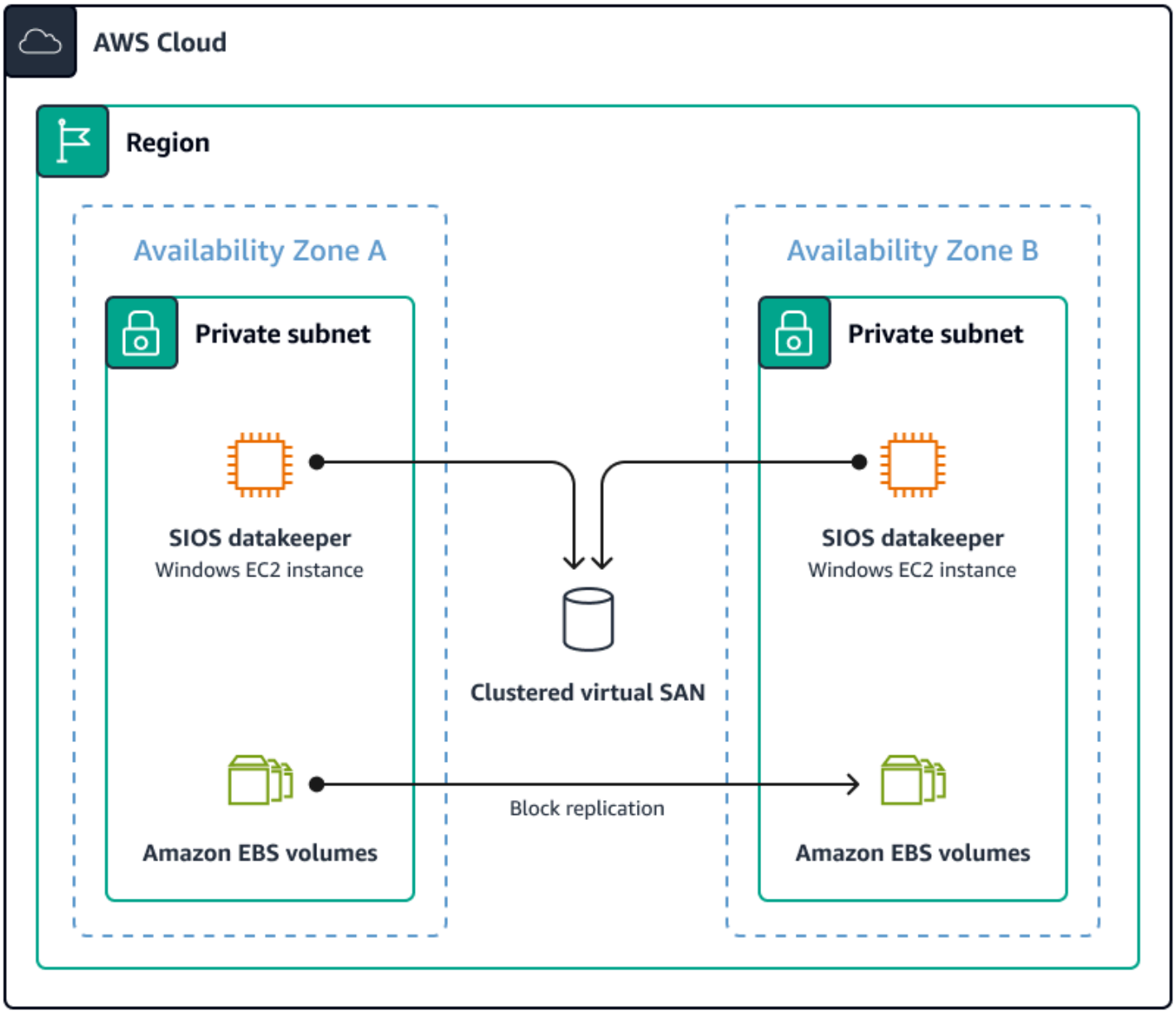
SIOS DataKeeper を使用して、アベイラビリティーゾーンとリージョンの両方FCIをカバーするSQLサーバーを作成し、コストを 58～71% 削減できます。SIOS DataKeeper は、 の高可用性のメリットを達成するのに役立ちますFCI。これによりSIOS DataKeeper 、組織にとってコスト効率と信頼性に優れたソリューションになります。

を使用する利点として、次の点を考慮してくださいSIOS DataKeeper。

- SIOS DataKeeper は、ローカルEBSボリュームSANを使用してクラスター化された仮想を作成し、可用性を高めるために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ます。ディザスタリカバリの場合、 SIOS DataKeeper はリージョン間で非同期レプリケーションを使用します。
- SIOS DataKeeper は、SQLServer Standard Edition を使用してエンタープライズクラスのクラスター機能を提供します。これにより、SQLServer Enterprise Edition を使用する SQL Server Always On 可用性グループで高可用性を実装するのと比較して、SQLサーバーのライセンスコストが 65～75% 削減されます。を使用するとSIOS DataKeeper、組織のニーズを満たす、可用性、柔軟性、コスト効率の高いSQLサーバー環境を作成できます。

> **Note**
>
> SQL Server Editions のコスト差の詳細については、このガイドの「Compare SQL Server Editions」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FCIを使用するSQLサーバーのアーキテクチャの例を示しています。

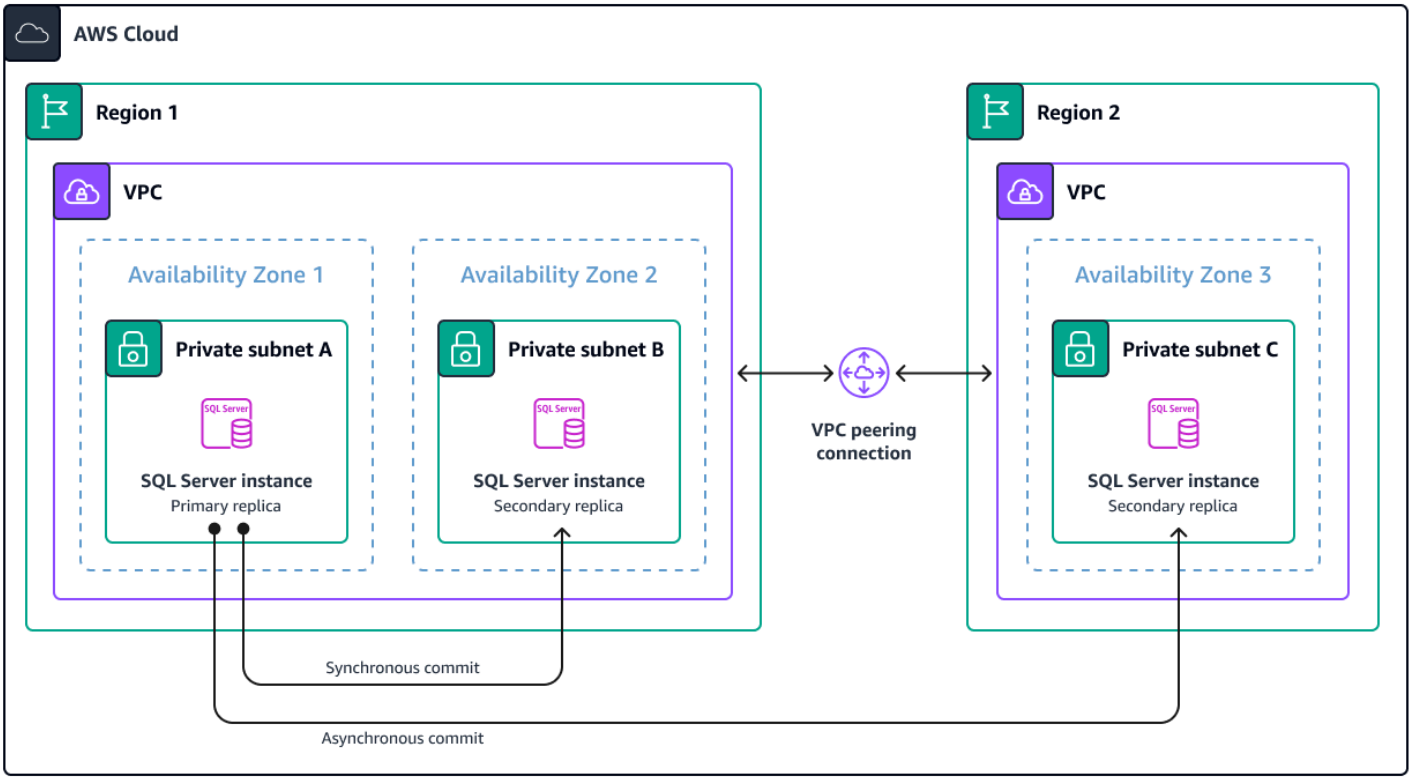


Always On 可用性グループ

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。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SQLサーバーを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ます。これには、可用性グループの手動フェイルオーバーが必要です。



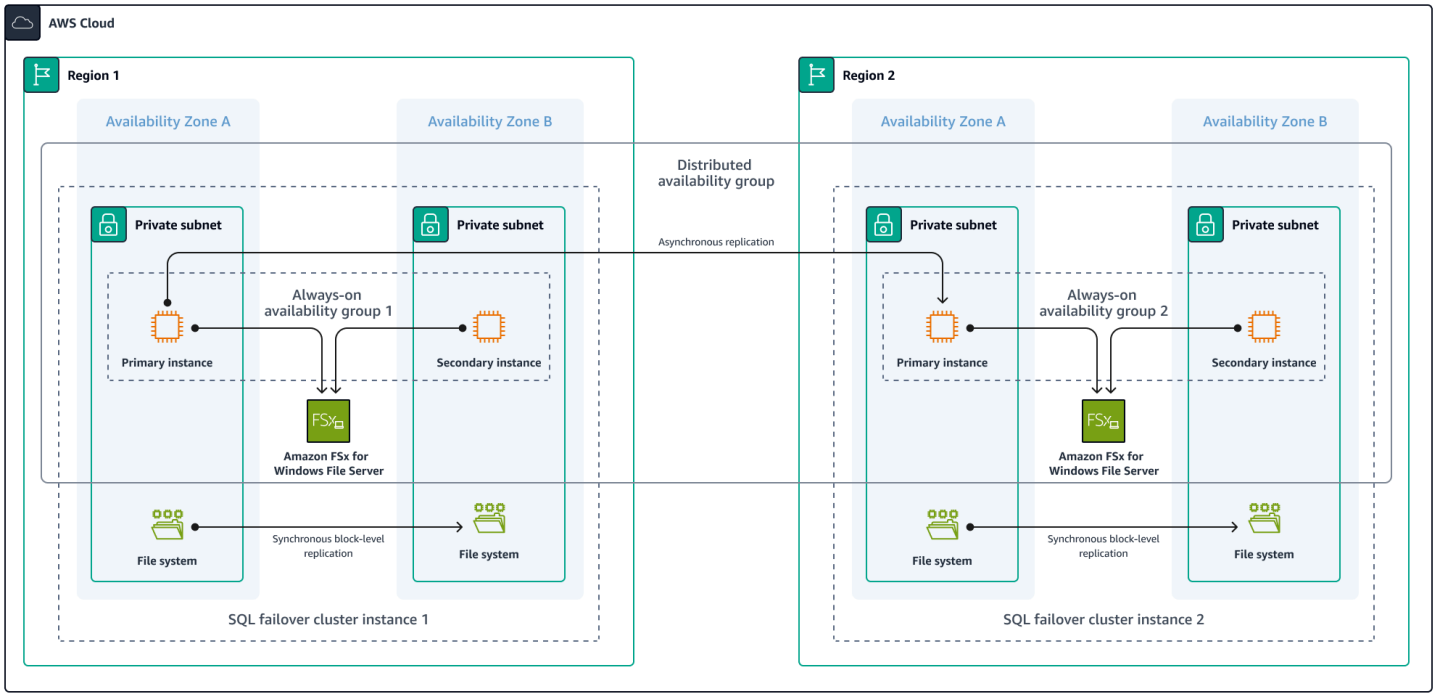
分散可用性グループ

信頼性やディザスタリカバリに妥協できないミッションクリティカルなSQLサーバーデプロイの場合は、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、事業継続性を維持し、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のレプリケーション、SQLサーバーなど、Amazon FSx for Windows File Server の機能を最大限に活用しますFCIs。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のSQLサーバー環境を作成できます。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冗長システムが得られます。このソリューションを際立たせるのは、ソリューション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は異なる Active Directory ドメインに参加できます。一方、証明書ベースの認証により、SQLサーバー環境を最大限に保護し、マルチリージョン DR 戦略に高いRTO要件とRPO要件を提供します。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、 AWS 「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。

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

ログ配送は、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送を使用してデータを保護してきました。

に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 を数分RTOで実現できます。万が一リージョンにアクセスできなくなった場合、ログの配送によりデータの安全性と復旧性が維持されます。

ログ配送を使用するその他の利点として、次の点を考慮してください。

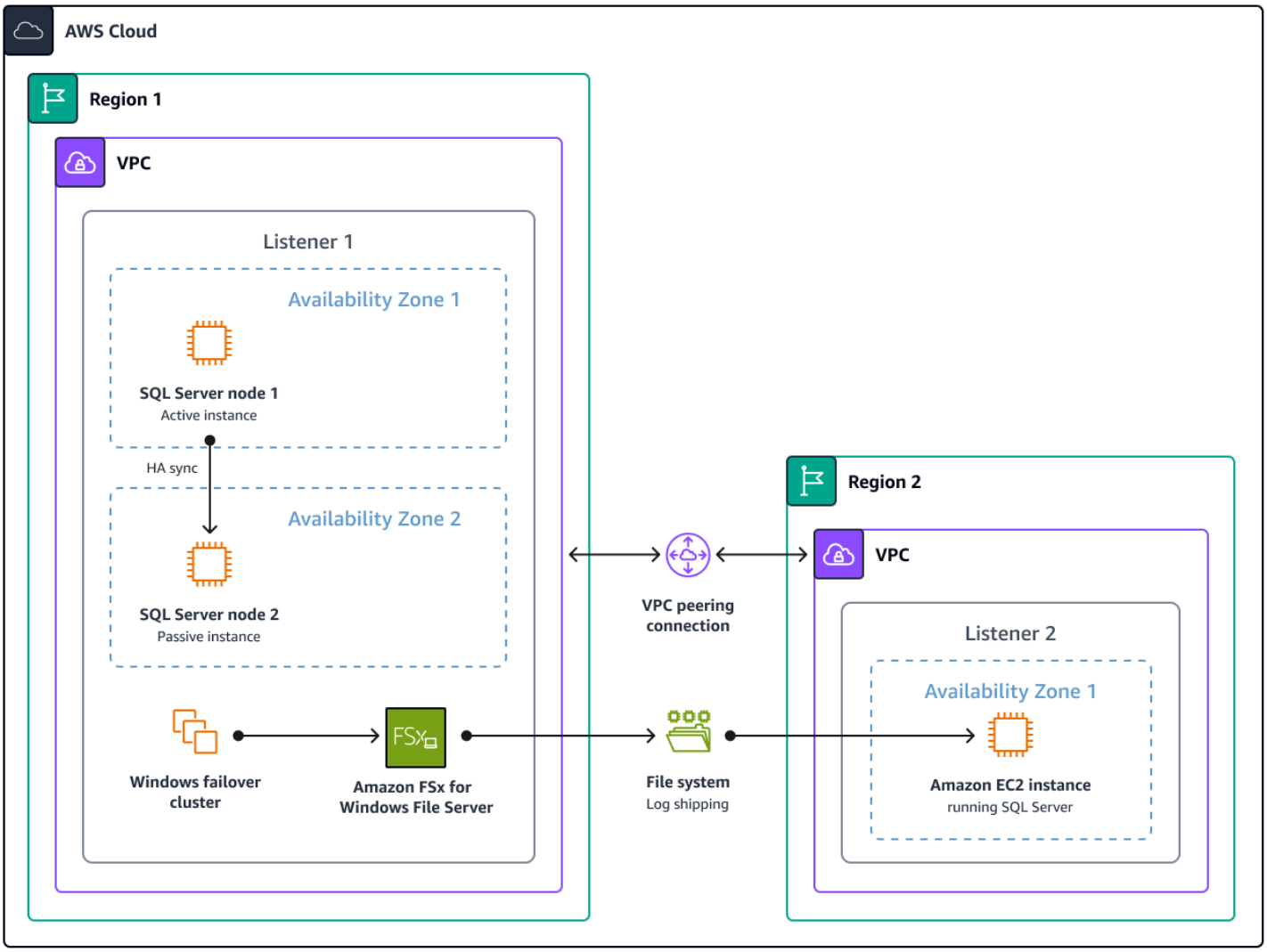
- リージョン間のディザスタリカバリの回復力にログ配送を使用することで、コストを削減し、ビジネス要件を満たします。ログ配送では、SQLサーバースタンダードエディションまたはSQLサーバーウェブエディションライセンスのみが必要なTCOため、 が削減されます。

- アクティブな Software Assurance でログ配送を使用して、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除します。Software Assurance でログ配送を使用する場合、プライマリ/アクティブSQLサーバー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設定する必要がなくなるため、SQLサーバーのライセンスコストを 65～75% 削減できます。これを行うには、SQLServer Standard Edition と SQL Server をログ配送FCIsと組み合わせて、ディザスタリカバリ要件を満たす必要があります。

> **Note**
>
> SQL Server Edition 間のコスト差の詳細については、このガイドの「Compare SQL Server Editions」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FCIで SQL Server のログ配送を使用して SQL Server DR を拡張する」を参照してください。

次の図は、ログ配送ソリューションのアーキテクチャの例を示しています。

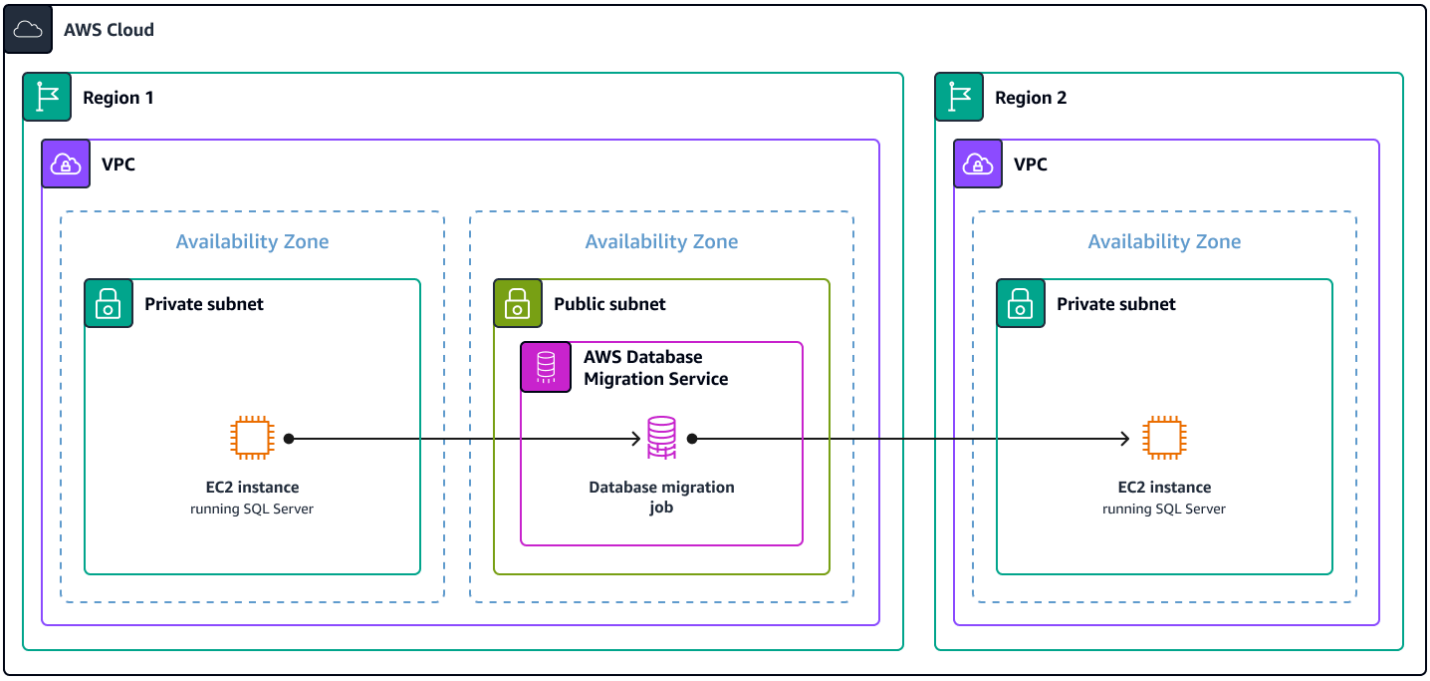


AWS Database Migration Service

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サーバー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、リソースの使用を最適化しながらインフラストラクチャへの投資 AWS 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、多額の追加コストを被ることなく、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、データが利用可能でアクセス可能であることを確認することができます。

次の図は、 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 DMS。



AWS Elastic Disaster Recovery

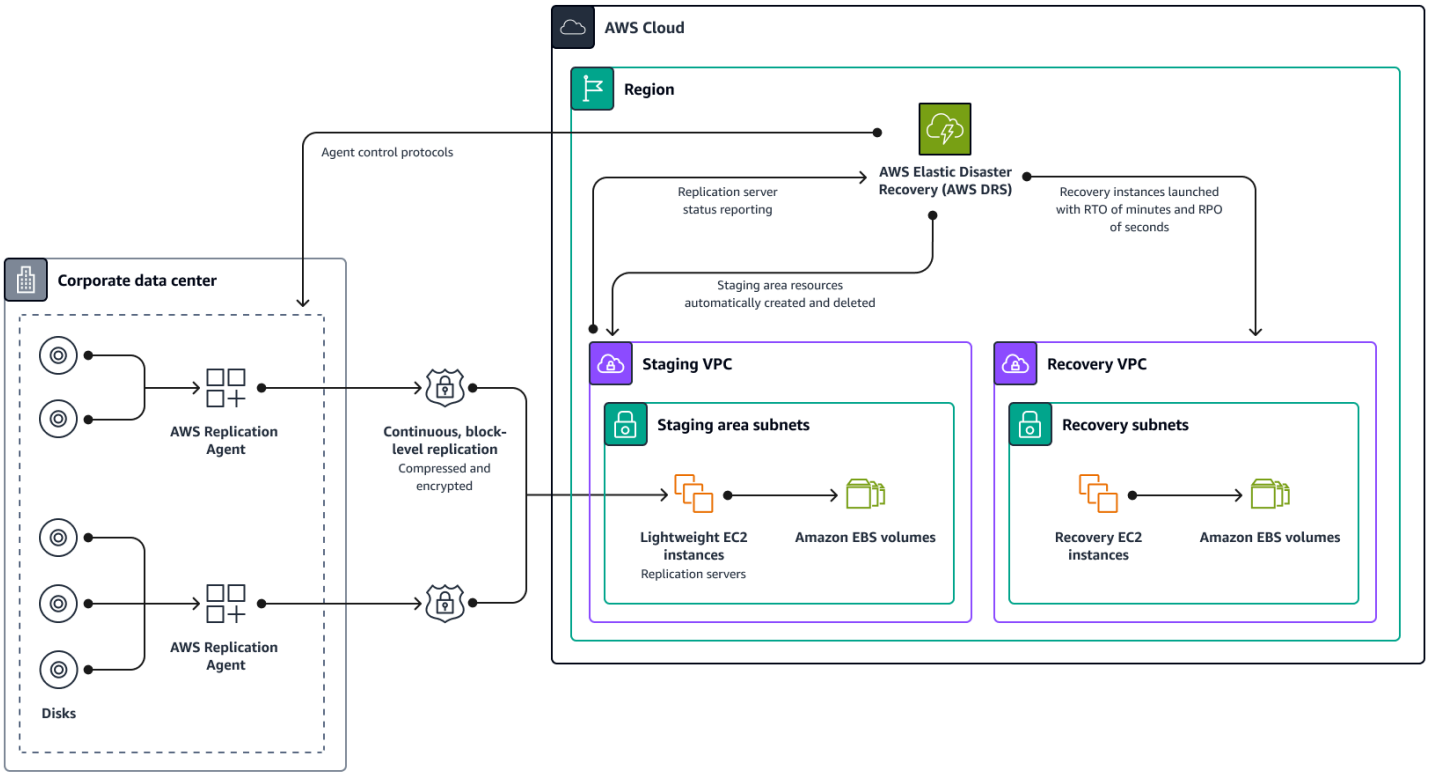
一部の組織では、すべての重要なビジネスアプリケーションにディザスタリカバリプランが設定されていることを確認する必要があります。以前は、これらの組織の多くは従来のディザスタリカバリソリューションに多額の投資を行っていました。そのためには、重複するインフラストラクチャ全体を事前に構築して維持する必要があります。このアプローチはコストがかかり、時間がかかり、スケーリングが困難です。

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、使用料は必要なときにのみ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アクティビティに集中できますが、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ケアします。

Server ではSQL、コスト効率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。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、耐障害性が高く可用性の高いSQLサーバーアーキテクチャのパッシブノードのライセンスがカバーされます。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依然として支払われています。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する必要もなく、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なく、DR 環境にレプリケートできます。この節約を組み合わせることで、SQLサーバー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例を示しています。



詳細については、 AWS ブログの Microsoft Workloads で を使用して復元された DR サイトの 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、次の前提が考慮されます。

- インスタンスタイプ – r5d.xlarge
- ライセンスタイプ – Windows と SQL Server の両方にライセンスが含まれています

- [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	コスト
ログ配布	不可	はい	はい	可能	SQL Server Enterprise Edition: $32,674.8 (2 ノード) SQL Server Standard Edition: $14,804.4 (2 ノード)
Always On 可用性グループ	あり	はい	可能	はい、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 (2 ノード) SQL Server Standard Edition: $14,804.4 (2 ノード)
常時オン FCIs	可能	いいえ	可能	はい (2 ノード)	SQL Server Standard Edition: $14,804.4
分散可用性グループ	あり	はい	はい	不可	SQL Server Enterprise Edition:

ソリューション	高可用性	ディザスタリカバリ	Enterprise Edition	Standard Edition	コスト
					$65,349.6 (4 ノード)
Elastic Disaster Recovery	不可	はい	はい	可能	約 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に 107.48 USD/月 注: Elastic Disaster Recovery は、レプリケートサーバーごとに時間単位で請求されます。コストは、ディスクの数、ストレージのサイズ、ドリルまたはリカバリの起動の数、レプリケートするリージョンに関係なく同じです。

ソリューション	高可用性	ディザスタリカバリ	Enterprise Edition	Standard Edition	コスト
SIOS データキーパー	あり	はい	はい	可能	Software Assurance を使用した Always On 可用性グループ (2 ノード、24 コア): $213,480 SIOS DataKeeper および ソフトウェアアシュアランスによる SQL Server Standard Edition で実行されている 2 ノード SQLサーバークラスター: $61,530 (2 ノード)
AWS DMS	不可	はい	はい	可能	r5.xlarge インスタンスと 1 TB のストレージに対して 745.38 USD/月

コスト最適化に関する推奨事項

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「サーバー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。
- 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とスループット要件を決定します。
  - IOPS = ディスク reads/sec + disk writes/sec
  - スループット = ディスク読み取り bytes/sec + disk write bytes/sec
- 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。
  - NVMe tempdbおよびバッファプール拡張機能のインスタンスストレージ
  - データベースファイルの io2 ボリューム
- Amazon 上の SQL Server のコスト最適化に関する推奨事項AWS Trusted Advisorには、 を使用しますEC2。SQL サーバー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、エディションなどの Amazon EC2 SQL Server ライセンスに含まれるインスタンス設定 Trusted Advisor を検査します。次に、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 のEBS正しいサイジングのレコメンデーションの両方 AWS Compute Optimizer に使用します。
- を使用してAWS Pricing Calculator、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**
>
> Side-by-side ライセンス込みEC2インスタンスを使用する場合、SQLサーバーエディションの変更には移行が必要です。

- 半年ごとまたは年に 1 回のディザスタリカバリドリルを実行して、 RTOと を定義してデータベースを復旧できる設計をより適切に設計しますRPO。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。

追加リソース

- Amazon FSx for Windows File SQL Server を使用して Microsoft 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および分散アベイラビリティグループを使用したSQLサーバー用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)
- でSQLサーバーのディザスタリカバリを設計する AWS: パート 1 (AWS データベースブログ)
- Amazon FSx for Windows での Microsoft SQL の高可用性 (YouTube)
- Amazon による Microsoft SQL Server のパフォーマンスの最大化 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)
- データセンターを NAS 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サーバーデプロイのコストの最適化 AWS (AWS ストレージブログ)
- (の Microsoft ワークロード) を使用してSQLサーバー Always On アベイラビリティーグループのディザスタリカバリを設定する方法 AWS Elastic Disaster Recovery AWS
- を使用して復元された DR サイトに SQL Server の高可用性を設定する方法 AWS Elastic Disaster Recovery (の Microsoft ワークロード AWS)

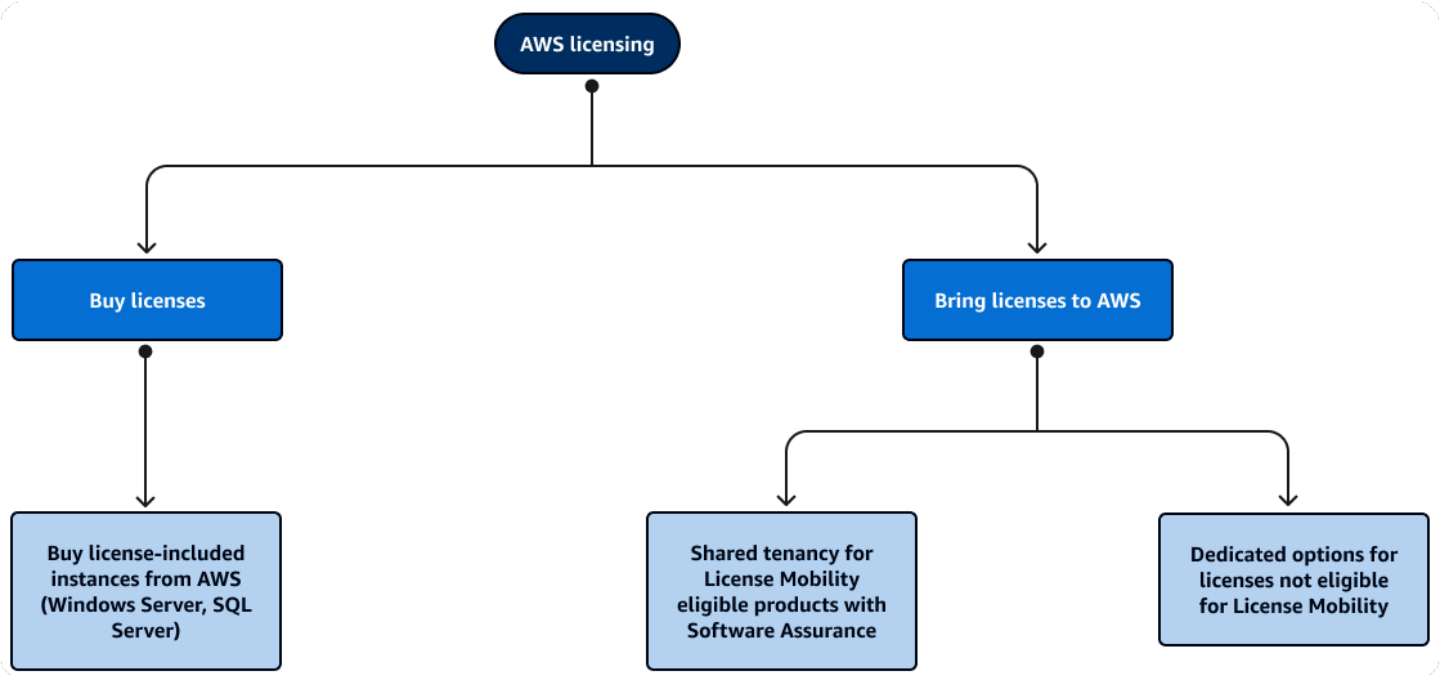
SQL サーバーライセンスを理解する

概要

ワークロードをクラウドに移行する企業が増えているため、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、 での 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サーバーの 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 では、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が用意されています。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 つの主要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では、使用分に対してのみ料金を支払い、オンデマンドでライセンスを購入して使用できます。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、SQLServer、およびその他の Microsoft 製品から選択できます。
2. Bring Your Own License (BYOL) 製品にライセンスモビリティ – このライセンスオプションは、既存のライセンスがあり、クラウドで使用したいシナリオ向けに設計されています。 では、Microsoft の License Mobility 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 AWS ことができます。ソフトウェアアシュアランス付きSQLサーバー (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み、 AWS インスタンスコストを削減できます。
3. BYOL ライセンスモビリティのない製品 – 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、専用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を 50% 以上節約できます。専有ホストは、ほとんどの場合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。

ライセンスの導入によるコストへの影響

ライセンスの持ち込みは、 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 AWS。独自のライセンスをお持ちの場合は、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/月 (ライセンス込み)
- Windows Server のみ: $259/月 (ライセンス込み)
- コンピューティングのみ (Linux): $127/月

最終的に、独自のライセンスを導入すると、 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 AWS。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 のサイズを適切に調整することです。適切なサイジングには、、メモリCPU、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するコアの数は、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調整することで、Microsoft ソフトウェアが実行す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数の両方を削減できます。

コスト最適化に関する推奨事項

ライセンスの最適化は、 でのコスト最適化の重要な要素です AWS。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大限の価値を実現できます。このセクションでは、ライセンス最適化のため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

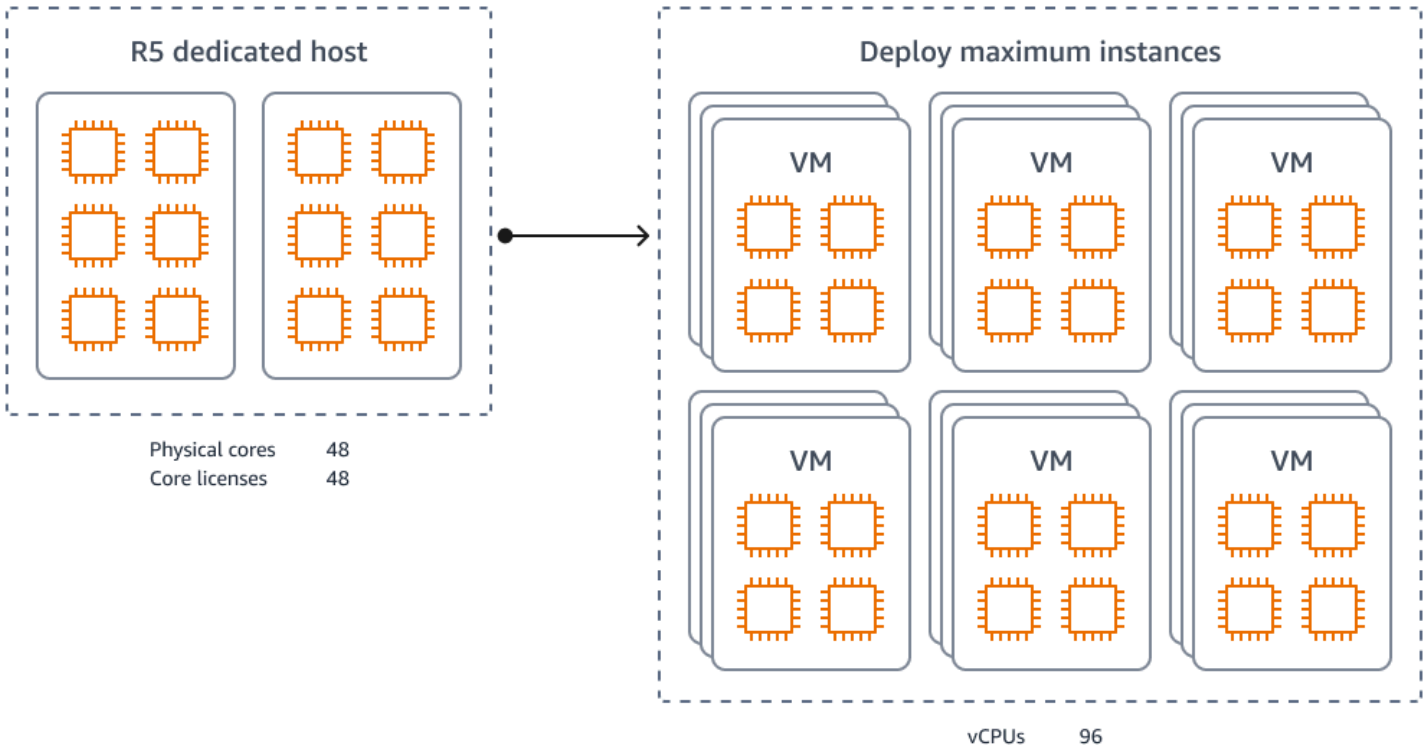
独自の Windows Server ライセンスの持ち込みは、ライセンス最適化の最も効果的な戦略の 1 つです。この戦略により、既存の投資を活用して AWS 支出を減らすことができます。

例えば、1/10/2019 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アクティブなエンタープライズ契約の下でライセンスを補正発注として購入した場合、Windows Server 2019 以前のバージョンを Amazon EC2 Dedicated Hosts 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、リストプロバイダー (Alibaba AWS、Google Cloud など) にデプロイしたときに変更したことに基づいています。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に含まれる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、それらの Windows Server ライセンスを Amazon EC2 Dedicated Hosts にデプロイすることはできます。

物理レベルのライセンス

物理コアレベルでのライセンスにより、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 コアの R5 専用ホストを考えてみましょう。これは 96 に変換されます vCPUs。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vCPUs、最大 96 のインスタンスの組み合わせをデプロイできます。



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い場合があります。物理コアレベルでライセンスすることで、インスタンスごとに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可能な限り最高の価値を実現できます。

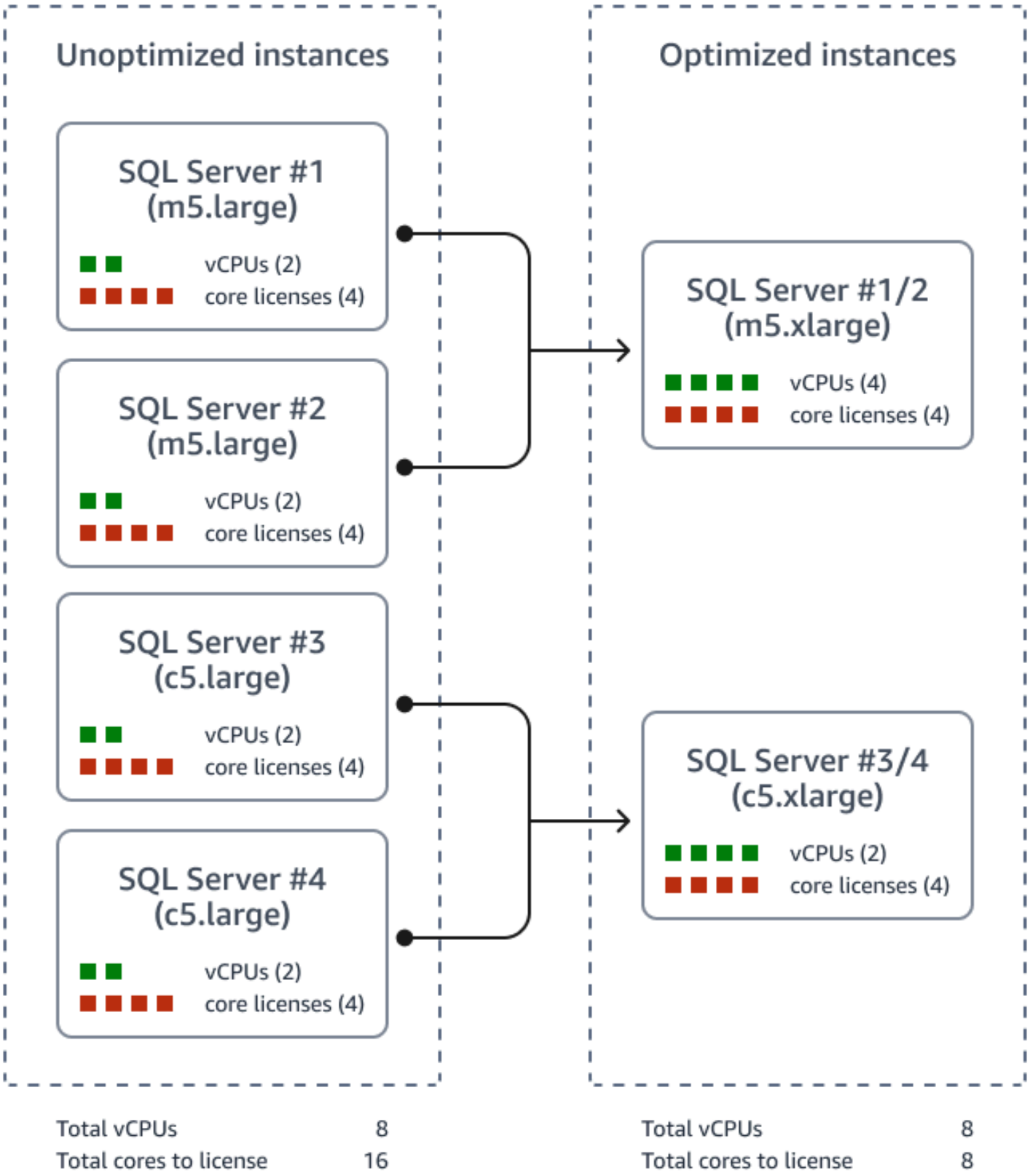
SQL サーバーの物理コアレベルでのライセンス

共有テナンシーでは、SQLサーバーライセンスはインスタンスvCPUsに割り当てられた数に基づきます。これとは対照的に、専用ホストでは、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。

R5 専用ホストの前の例と同様に、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。これとは対照的に、v によるライセンスのみのオプションである共有テナンシーではCPU、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、専用ホストは、共有テナンシーと比較して、SQLサーバー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 ライセンスを導入することで、インスタンスのコストを削減することに加えて行われます。

SQL サーバーインスタンスの統合

SQL サーバー統合は、複数のSQLサーバーインスタンスを 1 つのサーバーに結合するプロセスです。SQL インスタンスに が 2 つしかない場合でも、サーバーにはインスタンスごとに少なくとも 4 つのコアライセンスが必要ですvCPUs。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の過剰ライセンスが発生し、必要以上に多くのライセンスを使用する可能性があります。


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、2 つのインスタンスを vCPUs 2 つのインスタンスをそれぞれ 4 つの 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 vCPUs できます。これは、8 ではなく 4 つのコアライセンスのみが必要なためです。

統合の詳細については、このガイドのSQLサーバー統合セクションを参照してください。

Server SQL 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です。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、SQLServer Enterprise Edition の 2 つの一般的な機能です。ただし、これらの機能には費用対効果の高い代替手段があり、SQLServer Enterprise Edition の全機能セットを必要としない場合に検討できます。例えば、SQLServer 2019 から SQL Server Standard Edition TDEを取得できます。Always On 可用性グループの代わりに、SQLServer Standard Edition で高可用性を実現するために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ーを使用できます。

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 Storage Blog の記事の「高可用性SQLサーバーデプロイのコストの最適化 AWS」を参照してください。

ライセンスコストを削減するだけでなく、SQLサーバーエディションをダウングレードすることで、ソフトウェアアシュアランスの支出を削減し、将来の調整を回避できます。未使用のライセンスをシェルフに返すと、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大限の価値を得ることができます。

SQL サーバー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、SQLServer Enterprise エディション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「サーバー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 SQL Server Developer Edition を使用する

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SQL、Enterprise や Standard Edition などのライセンス可能な Server 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、 に移行すると AWS、これらのライセンスを に引き渡すことはできません。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、無料で利用できる SQL Server のフル機能のエディションです。このエディションは、SQLサーバーバージョン 2016 以降で使用できます。Microsoft ウェブサイトからダウンロードできます。SQL Server Developer Edition は、本番稼働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、本番稼働以外のすべての環境で使用することを意図しています。

Server SQL Developer Edition を非本番環境で使用すると、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「サーバー開発者エディションの評価」セクションを参照してください。

SQL サーバーワークロードCPUの最適化

場合によっては、 RAM やネットワークの制限など、他の要因により、ワークロードに必要なCPUs数を超えるインスタンスタイプを選択する必要が生じることがあります。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 コアライセンスをお持ちのほとんどのお客様と同様に、ハイパースレッディングを無効にするか、EC2インスタンスCPUsをオフにして、ホストCPUsが利用できる数を制限できます。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削減しながらRAM、 などの他のインスタンス機能を利用できます。

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

インスタンスタイプ	合計 vCPUs	最適化CPUs機能を備えたアクティブ vCPU	SQL サーバーライセンスの節約
r5.4xlarge	16	8	50%

インスタンスタイプ	合計 vCPUs	最適化CPUs機能を備えたアクティブ vCPU	SQL サーバーライセンスの節約
r5.12xlarge	48	8	83%

インスタンスのサイズを適正化すれば、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使用していることを確認することができます。が新しいインスタンスタイプ AWS を導入するにつれて、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。

追加リソース

- Amazon Web Services と Microsoft: よくある質問 (AWS ドキュメント)

SQL サーバー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、まずこのガイドのSQL「サーバーのライセンスを理解し、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」セクションを読む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、その経験を活かし、最小限の仕様から高性能のマルチリージョンクラスターまで、サーバーSQL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。

Server の正しいEC2インスタンスの選択SQL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、SQLおよび Server の機能が Amazon EC2製品とどのように連携するかを理解することは、アプリケーションに最適なEC2インスタンスに導くのに役立ちます。

このセクションでは、さまざまなSQLサーバーワークロードと、それらと特定のEC2インスタンスを組み合わせてライセンスコストとコンピューティングコストを最小限に抑える方法について説明します。

コスト比較

Amazon EC2 では、Windows Server および SQL Server のライセンスに応じて、独自のライセンス (BYOL) を持ち込み、または支払いを行うことができます。ライセンスの場合 pay-as-you-go、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ベイクされます。例えば、異なる料金AMIsで異なる を持つことができます。の料金はAMI、 がAMI実行する SQL Server エディションによって異なります。

Windows Server と SQL Server の料金は明細化されません。などのツールでは項目別料金はありませんAWS Pricing Calculator。ライセンス込みオファリングのさまざまな組み合わせを選択した場合、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 ドル	-	-	183.96 ドル
r5.xlarge	Linux + SQL デベロッパー	183.96 ドル	$0	$0	183.96 ドル
r5.xlarge	Windows Server (LI)	183.96 ドル	$134.32	-	$318.28
r5.xlarge	Windows + SQL デベロッパー	183.96 ドル	$134.32	$0	$318.28
r5.xlarge	Windows + SQL ウェブ (LI)	183.96 ドル	$134.32	49.64 ドル	$367.92
r5.xlarge	Windows + SQL標準 (LI)	183.96 ドル	$134.32	$350.4	$668.68
r5.xlarge	Windows + SQL	183.96 ドル	$134.32	1,095 ドル	1413.28 ドル

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
	Enterprise (LI)				

> **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することです。詳細については、このガイドのSQL「サーバー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サーバーのライセンスコストの 7 倍以上、Standard Edition から Enterprise Edition への移行コストの 3 倍以上になります。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サーバーのパフォーマンスを改善しようとしているシナリオの例を考えてみましょう。MACO エキスパート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してSQLサーバー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サーバーのライセンスを 8 から 4 vCPUs に減らすことができましたvCPUs。

最適化前：

サーバー	EC2 インスタンス	SQL サーバーエディション	月額コスト
製品DB1	x1e.2xlarge	エンタープライズ	$3,918.64
製品DB2	x1e.2xlarge	エンタープライズ	$3,918.64
合計			$7,837.28

最適化後：

サーバー	EC2 インスタンス	SQL サーバーエディション	月額コスト
製品DB1	x2iedn.xlarge	標準	$1,215.00
製品DB2	x2iedn.xlarge	標準	$1,215.00
合計			$2,430.00

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月額 5,407 ドルを節約できました。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。

コスト最適化に関する推奨事項

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、メモリへの依存度を理解することです。SQL サーバーは、オペレーティングシステムで使用RAMされていない使用可能なすべての 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。メモリ内のデータの操作は、ディスクからデータを常にプルし、変更を加え、ディスクに書き戻すよりもはるかに効率的です。代わりに、SQLサーバー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 に保持しますRAM。データに加えられた変更はメモリ内で発生し、後でディスクに強化されます。

> **Note**
>
> Server の変更のSQL記述方法の詳細については、Microsoft ドキュメントの「ページの書き込み」を参照してください。

SQL Server は大量の でパフォーマンスが向上するためRAM、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 1 対 8 vCPU-to-RAM の比率で、インテルプロセッサ、AMDプロセッサ、拡張ネットワーク、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には、同じオプションを多数組み合わせ、比率を vCPU-to-RAM 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高いため、あらゆる形状とサイズのSQLサーバーワークロードに適用できます。

最小リソース (4 未満vCPUs) 未満のワークロード

一部のユースケースはバースト性 (T3) インスタンスでうまく機能しますが、通常はSQLサーバーワークロードにバースト性インスタンスを使用しないことをお勧めします。SQL サーバーのライセンスは、インスタンス vCPUs に割り当てられた の数に基づいています。SQL サーバーが 1 日の大半アイドル状態であり、バーストクレジットを取得している場合は、十分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サーバー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 vCPUs 分のコンピューティングパワーを必要としないSQLサーバーワークロードがある場合、使用していないSQLサーバーライセンスに支払いが行われます。これらのシナリオでは、複数のSQLサーバーインスタンスをより大きなサーバーに統合するのが最善です。

最小リソース (64 GB 未満RAM) を使用したワークロード

64 GB 未満のSQLサーバーワークロードの多くは、高性能または高可用性RAM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、SQLServer 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 は、パブリックおよびインターネットにアクセスできるウェブページ、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するためにのみ使用できます。アプリケーション (顧客関係管理、エンタープライズリソース管理、その他の類似アプリケーションなど) のサポート line-of-businessには使用できません。

SQL Server Web Edition は最大 32 GB vCPUs と 64 GB まで拡張RAMでき、SQLServer 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 のワークロードよりも

10% 安価なコンピューティング料金の r6a のような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コストとSQLライセンスコストを最小限に抑える良い方法です。

平均リソースを持つワークロード (128 GB 未満RAM)

SQL Server Standard Edition は、最大 128 GB のSQLサーバーワークロードの大部分で使用されますRAM。SQL Server Standard Edition は SQL Server Enterprise Edition よりも 65～75% 安価で、最大 48 GB vCPUs と 128 GB まで拡張できますRAM。通常、128 GB RAMの制限は 48 vCPU の制限より前に適用されるため、SQLServer 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です。

SQL サーバーには、バッファプール拡張機能と呼ばれる機能があります。この機能を使用すると、SQLサーバーはディスクの一部を使用して の拡張機能として機能しますRAM。バッファプール拡張機能は、Amazon インスタンスストレージ NVMeSSDsで使用される のように、超高速ストレージと組み合わせるとうまく機能します。 EC2 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 に代わるものではありませんRAM。ただし、128 GB を超えるが必要な場合はRAM、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。

高性能ワークロード (128 GB 以上RAM)

SQL 高いパフォーマンスを必要とするサーバーワークロードは、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、間違った選択ができなくなる可能性があります。

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 Intel Xeon プロセッサー	3.5 GHz 第 3 世代 Intel Xeon プロセッサー	3.9 GHz 第 4 世代 Intel Xeon スケーラブルプロセッサ	3.5 GHz 第 3 世代 Intel Xeon プロセッサー	4.5 GHz 第 2 世代 Intel Xeon プロセッサー
CPU：RAM 比	1:8	1:8	1:8	1:32	1:32

	r5b	r6idn	r7iz	x2iedn	x2iezn
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	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ことは、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EBS高性能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 が優先されます。これは、io2 Block Express ボリュームを使用し、最大ストレージ IOPS 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、EBS高性能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 EBS のパフォーマンス強化は、r6idn でさらに一歩進み、最大 350,000 IOPSまで引き上げられます。r6idn には、tempdb と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、SQL サーバーのパフォーマンスをさらに向上させます。

- x2iedn – x2iedn は r6idn に似ています。拡張された EBS、拡張ネットワーキング、NVMeSSDインスタンスストレージのレベルは同じですが、メモリ負荷が高く、CPU数量が少ない (SQLサーバーライセンスコストが低い) 場合は 1:32 vCPU-to-RAM の比率です。
- x2iezn – x2iezn の「z」属性は、このインスタンスタイプがプロセッサのパフォーマンスが高いことに焦点を当てていることを示します。Cascade Lake プロセッサには、最大 4.5 のオールコアターボ周波数GHzがあります。このEC2インスタンスを 1:32 vCPU-to-RAM の比率と組み合わせて、vCPU 数量を低く抑えるシナリオで使用することをお勧めします。これにより、SQLサーバーのライセンスコストが低くなります。
- r7iz – r7iz の「z」属性は、このインスタンスタイプがプロセッサのパフォーマンスが高いことに焦点を当てていることを示します。Sapphire rapids プロセッサのオール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 1:8 vCPU-to-RAMの比率で、高頻度プロセッサのパフォーマンスを優先します。

追加リソース

- 汎用 Amazon EC2インスタンス (AWS ドキュメント)
- 比較ツール (Vantage)
- ライセンス – SQLサーバー (AWS ドキュメント)

インスタンスの統合

このセクションでは、複数のSQLサーバー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サーバーデータベースエンジンのインストールプロセスの一部です。SQL サーバーインスタンスは、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。インスタンスにはすべての独自のファイルとサービスがあるため、インスタンスが相互に干渉することなく、同じオペレーティングシステムに複数のSQLサーバー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サーバー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度に使用しないようにします。各SQLサーバーインスタ

ンスに独自のオペレーティングシステムと独自のリソースを提供することは、リソースガバナンスに依存するよりも優れた境界です。これは、大量の RAM と CPUリソースを必要とする高性能SQLサーバーワークロードに特に当てはまります。

ただし、すべてのSQLサーバー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サーバーインスタンスを割り当てます。通常アクティブではない小規模なクライアントまたはクライアントの場合、つまり、最小限のリソースでSQLサーバーインスタンスを実行することを意味します。

Microsoft SQL Server 2019: ライセンスガイド に記載されているように、SQLサーバーを実行している各サーバーは、少なくとも 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 のみのサーバーを実行してもvCPUs、4 つの のライセンスをSQLサーバーに付与する必要がありますvCPUs。SQL Server Standard Edition を使用する場合、Microsoft のパブリックSQLサーバー料金に基づくと、3,945 ドルの差があります。最小限のリソースを使用して単一のSQLサーバー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する必要が生じると、合計コストがかか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 つのサーバー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サーバーインスタンスを同時に実行する 1 つの大きな Windows Server SQL サーバーの実行の違いを比較するシナリオ例について説明します。

各SQLサーバーインスタンスに必要なのが 2 GB vCPUs と 8 GB のみの場合RAM、サーバーライセンスの合計コストは 7,890 USD SQL で、1 時間あたりのコンピューティングコストは 0.096 USD です。

EC2 インスタンス	vCPUs	RAM	価格	vCPUs からライセンス	SQL サーバーライセンスコストの合計
m6i.large	2	8	0.096	4	7,890 ドル

これを 4 つのサーバーに拡張すると、SQLサーバーライセンスの合計コストは 31,560 USD で、時間単位のコンピューティングコストは 0.384 USD になります。

EC2 インスタンス	vCPUs	RAM	価格	vCPUs からライセンス	SQL サーバーライセンスコストの合計
4x m6i.large	2	32	0.384	16	31,560 ドル

4 つのSQLサーバーインスタンスをすべて 1 つのEC2インスタンスに結合すると、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サーバー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ための総コストを 15,780 ドル削減できます。

EC2 インスタンス	vCPUs	RAM	価格	vCPUs からライセンス	SQL サーバーライセンスコストの合計
m6i.2xlarge	8	32	0.384	8	15,780 ドル

> **Note**
>
> 上記の表では、 `us-east-1`リージョンで Windows Server を実行している Amazon EC2 サーバーのコンピューティングコストは、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 のパブリックSQLサーバー料金 を指します。

コスト最適化に関する推奨事項

SQL サーバーインスタンスの統合を検討している場合、最大の懸念は、統合する各インスタンスのリソース消費です。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)、および SQL Server のネイティブモニタリングツールがあります。

SQL サーバー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 (CPU、メモリ、ネットワーク帯域幅) は何ですか？
- スパイク中に消費されるリソース (CPU、メモリ、ネットワーク帯域幅) は何ですか？
- スパイクはどのくらいの頻度で発生しますか？ スパイクは一貫していますか？
- あるサーバーのリソーススパイクは、別のサーバーのリソーススパイクと一致していますか？
- SQL Server が使用するストレージIOPSとスループットは何ですか？

SQL サーバーインスタンスを組み合わせる計画を進める場合は、 クラウドオペレーションと移行ブログの 1 つの Amazon EC2インスタンス投稿でSQLサーバーの複数のインスタンスを実行するを参照してください。 AWS この記事では、SQLサーバーで設定変更を行い、追加のインスタンスを追加する方法について説明します。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てい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追加のインスタンスはそれぞれ、「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。
- SQL サーバーブラウザを実行して、名前付きインスタンスへのトラフィックを調整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と個別のログインに別々の場所を使用できます。
- SQL Server Max 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ます。合計も、基盤となるオペレーティングシステムに十分なメモリを残します。
- Server のSQLネイティブバックアップおよび復元機能、またはAWS DMS移行または統合に使用できます。

追加リソース

- SQL サーバーライセンスデータシート (AWS クラウドオペレーションと移行ブログ)
- SQL Server Multiple instance Setup ブログ記事 (AWS Cloud Operations & Migrations ブログ)
- SQL サーバーのベストプラクティスガイド (AWS 規範ガイダンスドキュメント)

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 サーバーのライセンスコストは、ワークロードを実行するためにコンピューティングコストを超えて簡単に拡張できます。間違ったエディションを選択した場合、使用していない機能や必要のない機能に対して料金を支払うことができます。このセクションでは、機能や相対コストなど、次の SQL Server エディションを比較します。

- Enterprise – SQL Server Enterprise Edition は、高性能、無制限の仮想化、およびいくつか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とビジネスインテリジェンスを提供します。
- ウェブ – SQL Server Web Edition は、ウェブホストまたはウェブ付加価値プロバイダー () である企業に適していますVAPs。このエディションは、総所有コストが低く、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と管理機能を提供します。

> **Important**
>
> SQL Server Web Edition を使用して、パブリックおよびインターネットにアクセスできるウェブページ、ウェブサイト、ウェブアプリケーション、ウェブサービスのみをサポートできます。SQL Server Web Edition を使用してアプリケーション (顧客関係管理や 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 line-of-businessすることはできません。

- デベロッパー – 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 エディションを他のエディションに更新できます。

> **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 リセラーからSQLサーバーライセンスを購入し、ソフトウェアアシュアランス AWS を使用して に持ち込むことができます。または、Amazon EC2 にライセンスが組み込まれている pay-as-you-goモデルでSQLサーバーライセンスを使用できますAMIs。

Microsoft リセラーからSQLサーバーライセンスを購入する場合、コアライセンスは 2 個入りパックで販売されます。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、エンタープライズエディションとスタンダードエディションのコスト比較を示しています。

Version	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 ドル	3,945 ドル	74%
2019	13,748 ドル	3,586 ドル	74%

> **Note**
>
> 上記の表の料金は、Microsoft の SQL Server 2022 および SQL Server 2019 のパブリック料金に基づいています。

次のコスト比較は、ライセンス込み Amazon でさまざまなエディションのSQLサーバーEC2をホストしていることを示していますAMIs。この比較では、SQLサーバーは `us-east-1`リージョンの r6i.xlarge (4 v CPU) 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 サーバーライセンスコスト	合計
R6i .xlarge (Linux)	$183.96	–	–	$183.96
R6i .xlarge + Windows	$183.96	$134.32	–	$318.28

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 サーバーライセンスコスト	合計
R6i .xlarge + SQL Server Web Edition	$183.96	$134.32	49.35 ドル	$367.63
R6i .xlarge + SQL Server Standard Edition	$183.96	$134.32	$350.4	$668.68
R6i .xlarge + SQL Enterprise Edition	$183.96	$134.32	1,095 ドル	$1,413.28

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サーバー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サーバーライセンスのコストを比較したものです。

エディション	% の節約
エンタープライズと比較したスタンダード	68%
ウェブと標準の比較	86%
ウェブとエンタープライズの比較	95%

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

コスト最適化に関する推奨事項

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サーバーエディション全体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、SQLServer 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ています。

機能	Enterprise Edition	Standard Edition	ウェブエディション	Express エディション
Server Database Engine、SQLServer SQL 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容量	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 コアのうち、いずれか小さい方に制限	4 つのソケットまたは 16 コアのうち小さい方に制限	4 ソケットまたは 4 コアのうち、いずれか小さい方に制限
SQL Server Database Engine のインスタンスあたりのバッファプールの最大メモリ	オペレーティングシステムの最大数	128 GB	64 GB	1410 MB
SQL Server Database Engine のインスタンスあたりのバッファプール拡張の最大容量	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

機能	Enterprise Edition	Standard Edition	ウェブエディション	Express エディション
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
Columnstore 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 vCPUs) 未満で 64 GB 未満の を必要とする場合は RAM、SQLServer Web Edition から評価を開始できます。ワークロードに 64 GB を超えるメモリやその他の高可用性オプションが必要な場合は、SQLServer 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、パブリックおよびインターネットにアクセス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できますが、SQLServer Web Edition を使用してビジネスアプリケーションライ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 Licensing Support または Microsoft リセラーにお問い合わせください。

SQL Server Standard Edition は、最大 24 コア (48 vCPUs) 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、r6id インスタンスに存在するもののように、SQLServer Standard Edition がローカルEC2インスタンスストレージ を利用できるようにします。 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.htmlこれにより、メモリは最大メモリ設定の 4 倍まで拡張されます。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が Enterprise Edition にアップグレードする必要がなくなる可能性があります。

メモリ使用率は、バッファプールとページの平均余命カウンターでデータベースページを検索することで識別できます。ページの平均余命は、ディスクにフラッシュバックされるまでに、ページがメモリ内にある時間を示します。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ています。

機能	Enterprise Edition	Standard Edition	ウェブエディション	Express エディション
サーバーコアのサポート 1	あり	はい	はい	可能
ログ配布	あり	はい	はい	不可
データベースのミラーリング	可能	FULL 安全モード	証人としてのみ	証人としてのみ
バックアップ圧縮	あり	はい	いいえ	なし
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	不可	なし
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	不可	いいえ	なし
基本的な可用性グループ	不可	2 ノード	不可	なし
オンラインページとファイルの復元	可能	いいえ	いいえ	なし
オンラインインデックス作成	可能	いいえ	いいえ	なし

機能	Enterprise Edition	Standard Edition	ウェブエディション	Express エディション
オンラインスキーマの変更	可能	いいえ	いいえ	なし
高速リカバリ	可能	いいえ	いいえ	なし
ミラーリングされたバックアップ	可能	いいえ	いいえ	なし
ホット追加メモリと CPU	可能	いいえ	いいえ	なし
暗号化されたバックアップ	あり	はい	いいえ	なし
Microsoft Azure へのハイブリッドバックアップ (へのバックアップURL)	あり	はい	いいえ	なし
ディザスタリカバリ用のフェイルオーバーサーバー	あり	はい	いいえ	なし
高可用性のためのフェイルオーバーサーバー	あり	はい	いいえ	なし

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。機能の詳細なリストについては、Microsoft ドキュメントのSQL「Server 2019 のエディションとサポートされている機能」を参照してください。

機能	Enterprise Edition	Standard Edition	ウェブエディション	Express エディション
(パフォーマンス) リソースガバナー	可能	いいえ	いいえ	なし
(セキュリティ) 透過的なデータベース暗号化 (TDE)	あり	はい	はい	不可
(セキュリティ) 拡張キー管理 (EKM)	可能	いいえ	いいえ	なし
(レプリケーション) Oracle パブリケーション	可能	いいえ	いいえ	なし
(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	可能	いいえ	いいえ	なし
変更データキャプチャ	あり	はい	いいえ	なし

SQL サーバーデベロッパーエディション

開発、QA、テスト、ステージング、UAT環境などの本番稼働以外のワークロードはすべて、SQLServer Developer Edition を使用してSQLサーバーのライセンスコストを 100% 削減できます。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 には専用インフラストラ

クチャは必要ありません。詳細については、このガイドの SQL Server Developer Edition の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 または Standard エディション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どうかを確認することがベストプラクティスです。例えば、データベースがエンタープライズ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部としてのオンラインのインデックス再作成などSQL、T- ではキャプチャできない Enterprise Edition の機能があります。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

Server にライセンスを付与する方法によってSQL、エディションを切り替えるオプションが決まります。AMIsSQLサーバー を含む にはAMIs、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 に拘束されますAMI。AWS 請求コードを使用して、 に含まれるSQLサーバーバージョンを確認できますAMI。 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 Edition を変更しても、 に関連付けられた請求は変更されませんAMI。新しいエディションの SQL Server AMIを実行している新しいEC2インスタンスにデータベースを移行する必要があります。

独自のライセンスをお持ちの場合は、柔軟性が高まります。通常、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でも side-by-side、SQLサーバーをインストールし、インスタンス間でデータベースを移行できます。エディションのダウングレードの詳細については side-by-side、MSSQLTipsウェブサイトのSQL「サーバーでのエディションのアップグレードとダウングレード」を参照してください。

追加リソース

- SQL Server 2022 のエディションとサポートされている機能 (Microsoft Learn)
- sys.dm_db_persisted_sku_features (トランザクションSQL) (Microsoft Learn)
- どのバージョンのSQLサーバーを使用する必要がありますか？ (ブレントオザール無制限)

- AWS Pricing Calculator (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、SQLServer 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上のために使用されます。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。

- 本番稼働用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、ドキュメントの「Amazon Web Services と Microsoft: よくある質問」を参照してください。 AWS

コストへの影響

Server SQL Developer Edition を非本番環境のワークロードに使用すると、開発環境とテスト環境の現在のSQLサーバーライセンスコストの 100% を節約できます。

SQL サーバーバージョン	SQL Server Enterprise Edition (2 コアパック)	SQL Server Standard Edition (2 コアパック)	SQL サーバーデベロッパーエディション
2022	15,123 ドル	3,945 ドル	空き
2019	13,748 ドル	3,586 ドル	空き

> **Note**
>
> 上記の表の料金は、Microsoft の SQL Server 2022 および SQL Server 2019 のパブリック料金に基づいています。

次の表は、 us-east-2 リージョンで 4 で実行 vCPUs 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ています。これは、 の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 サーバーライセンス料金	合計料金
r5.xlarge	Linux (コンピューティング料金)	183.96 ドル	–	–	183.96 ドル
r5.xlarge	Linux + SQL Server Developer Edition	183.96 ドル	$0	$0	183.96 ドル
r5.xlarge	Windows Server (LI)	183.96 ドル	$134.32	–	$318.28
r5.xlarge	Windows + SQL Server デベロッパーエディション	183.96 ドル	$134.32	$0	$318.28
r5.xlarge	Windows + SQL Server Web Edition (LI)	183.96 ドル	$134.32	49.64 ドル	$367.92
r5.xlarge	Windows + SQL Server	183.96 ドル	$134.32	$350.4	$668.68

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 サーバーライセンス料金	合計料金
	Standard Edition (LI)				
r5.xlarge	Windows + SQL Server Enterprise Edition (LI)	183.96 ドル	$134.32	1,095 ドル	1413.28 ドル

コスト最適化シナリオ

データ整合性企業が新しい買収を行った後、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て、 の他のワークロードと統合したいと考えました AWS クラウド。初期料金では、会社のSQLサーバーワークロードは、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。は見積もりMACOSMEを評価し、お客様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でSQLサーバーライセンスの料金を支払っていることがわかりました。移行中に非本番環境ワークロードを SQL Server Developer Edition に切り替えることで、同社はSQLサーバーライセンスを 40% 削減しました。

SQL Amazon に含まれるサーバーライセンス EC2

ライセンス込み AMIs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。ライセンスに含まれるインスタンスのライセンスコストは に拘束されますAMI。Server SQL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インスタンスにはライセンスコストがかかります。

Developer Edition に変換するには、SQLServer 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サーバーデータベースを移行できます。詳細については、Microsoft SQL Server データベースのガイドへの移行の「サーバーデータベースの移行方法SQL AWS クラウド」を参照してください。Automated 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。

SQL Amazon BYOL上のサーバー EC2

を使用するSQLサーバーインスタンスがある場合はBYOL、次のインプレース変換または side-by-side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SQL「サーバー開発者デプロイの自動化」の記事を参照してください。
- Server SQL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したり、あるSQLインスタンスから別のインスタンスにデータベースをデタッチ/アタッチしたりします。
- 一括デプロイにはオートメーションツールを使用します。

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

追加リソース

- Server SQL Developer Edition をデプロイするための SQL Server Developer デプロイの自動化 EC2 (AWS ブログ)
- SQL 2022 年の料金 (Microsoft)
- SQL 2019 年の料金 (Microsoft)
- ライセンスオプション (SQLAmazon 上のサーバーEC2)
- AWS Pricing Calculator (SQLAmazon EC2ドキュメント上のサーバー)
- Microsoft SQL Server 2019 ライセンスガイド (Microsoft からダウンロード)
- SQL Server 2022 Developer Edition (Microsoft からダウンロード)

Linux でSQLサーバーを評価する

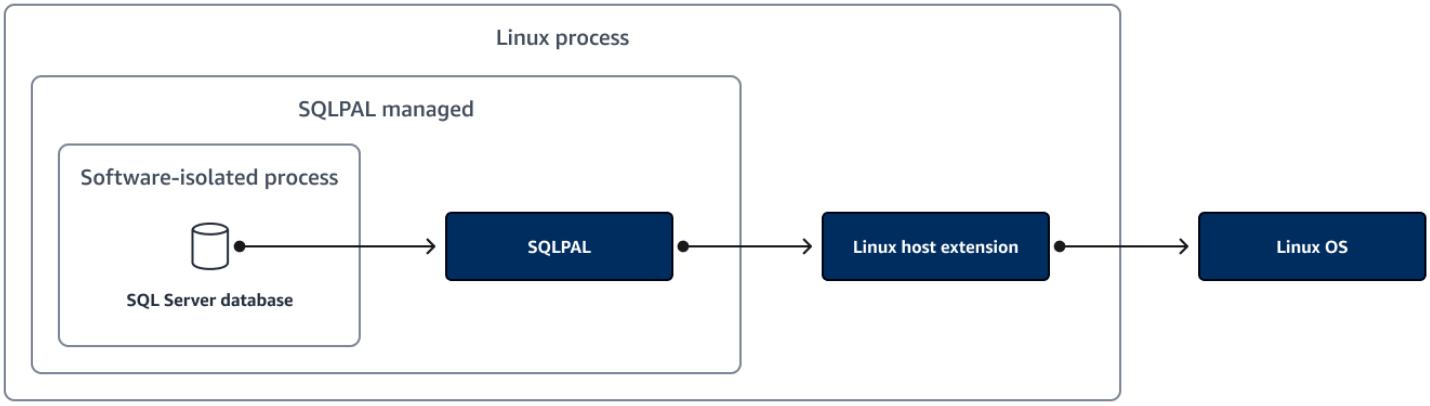
概要

SQL Server 2017 以降、Linux オペレーティングシステムに SQL Server をインストールできるようになりました。SQL Server on Linux はエンタープライズ対応で、柔軟性、高性能、セキュリティ機能、削減された TCO、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サーバーは Red Hat Enterprise Linux (RHEL ）、SUSELinux Enterprise Server (SLES ）、Ubuntu、および 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 と Linux の両方で同じように実行されますが、Linux を使用する際には特定のタスクにいくつかの基本的な変更があります。Linux と Windows で SQL Server Always On アプリケーションを実行する主な違いの 1 つは、フェイルオーバークラスター化に関連しています。Windows Server ホストに Always On 可用性グループをデプロイする場合、フェイルオーバークラスター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ー (WSFC) と Active Directory を活用できます。ただし、Linux でのフェイルオーバークラスター化をサポートするために WSFCも Active Directory も利用できません。Linux 上の SQL Server のフェイルオーバークラスターを起動する場合は、 AWS Launch Wizardを使用して、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インストールを簡素化できます。

SQL Windows と Linux 上のサーバーは共通のコードベースを共有します。つまり、SQLサーバーコアエンジンは Linux で実行するようにまったく変更されていません。SQL 次の図に示すように、サーバー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サーバー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、単にネイティブ 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使用状況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、 がロードされて初期化されSQLPAL、SQLサーバーが表示されます。SQLPAL は、残りのコードに必要な翻訳を提供する分離され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、オペレーティングシステムに関係なく、Windows で SQL Server を非常に強力にしたのと同じエンタープライズレベルのコア機能と利点を利用できます。

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 ドルです。この削減は、より安価なサーバーエディションの使用と比較して、SQLサーバーコストの合計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	エディション	Windows 上の SQL Server の月額コスト	Linux での SQL Server の月額コスト	削減量
r5.2xlarge	Web	$735	466 ドル	37%
r5.2xlarge	標準	1,337 ドル	1,068 ドル	20%
r5.2xlarge	エンタープライズ	2,826 ドル	2,558 ドル	10%

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

SMB セグメント内の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 SQL サーバーのセットで Server Developer Edition を既に使用しています。Windows with SQL Server Developer Edition から Linux with SQL Server Developer Edition に切り替えることで、ISVお客様は開発ワークロードを 33% 削減できます。次の表は、このシナリオの次の推定コストを示しています。

見積り	月額コスト
Windows + SQLサーバー	$9,307.72
Linux + SQLサーバー	$6,218.36
推定コスト削減額	$3,089.36 (33%)

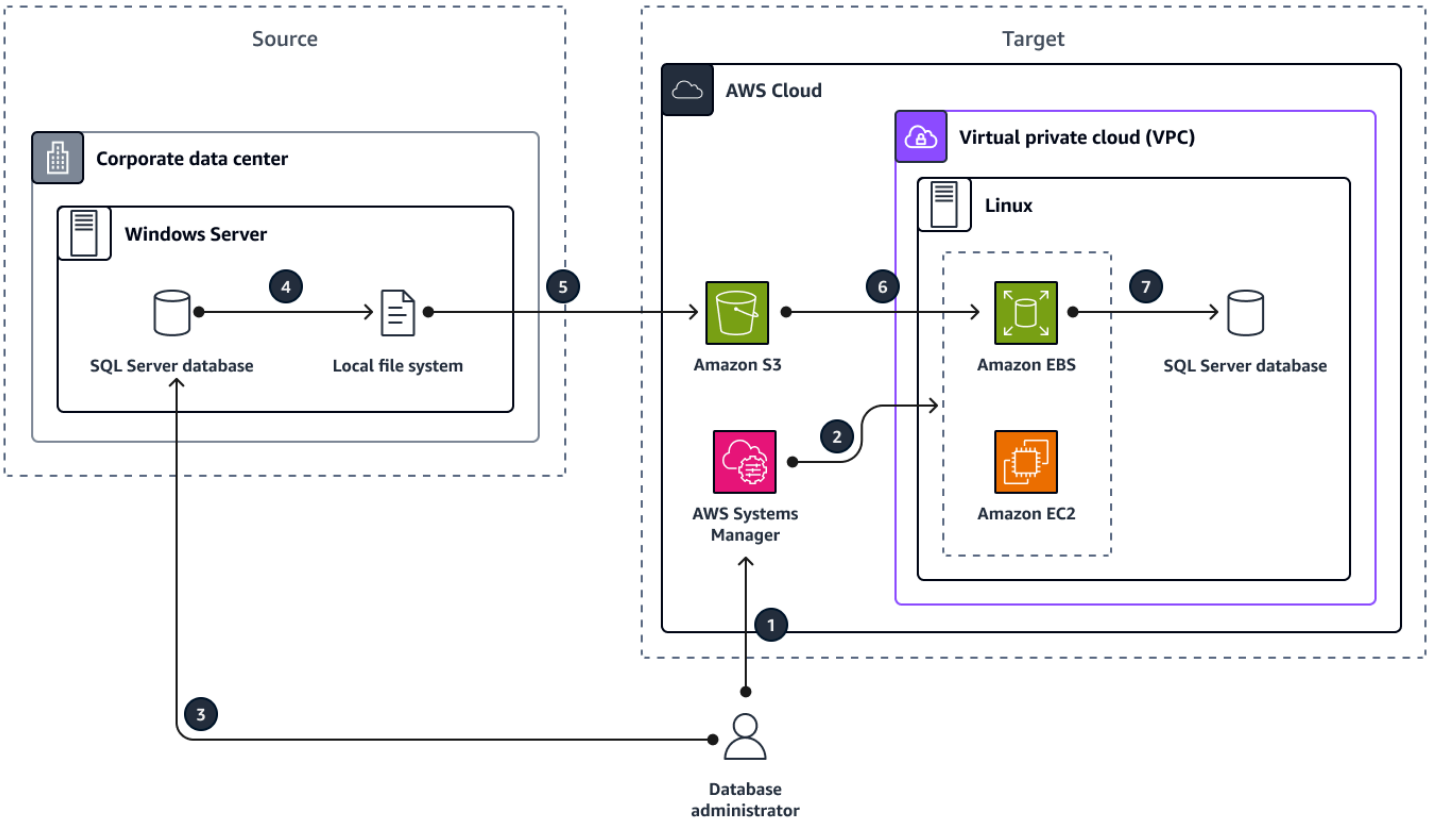
別のシナリオ例では、ある企業がライセンス込みSQLサーバーEC2インスタンスを Windows から Linux に移行します。同社は、Windows Server のライセンスコストを年間合計 300,000 ドル削減します。これは、請求総額 AWS の約 20% です。

コスト最適化に関する推奨事項

次の点を考慮することをお勧めします。

- SQL Linux 上の 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 Databases 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サーバーワークロードを Windows から Linux オペレーティングシステムに移行するのに役立つスクリプティングツールです。
- Server のバックアップおよび復元機能を使用して、Windows SQL 上の SQL Server から Linux に切り替えることもできます。
- を使用して、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AWS Launch Wizard。Launch Wizard は、アプリケーションのニーズに応じて、スタンドアロンシナリオと高可用性シナリオの両方で Linux または Ubuntu に SQL Server をデプロイできます。詳細については、 AWS ブログの Microsoft Workloads の投稿で SQL Server Always on Linux にデプロイ AWS Launch Wizardするを参照してください。

次の図は、Microsoft SQL Server Databases で 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。



追加リソース

- Linux 上の SQL Server の概要 (Microsoft Learn)
- SQL Server on Linux のインストールガイド (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 の高可用性SQLサーバー (AWS オープンソースブログ)

SQL サーバーバックアップ戦略の最適化

概要

ほとんどの組織は、復旧ポイント目標 (RPO) 、最後のバックアップからの最大許容時間、復旧時間目標 (RTO) 、サービスの中断とサービスの復旧の間の最大許容遅延の現在の要件を満たすために、Amazon EC2 上の SQL Server 上のデータを保護する適切なソリューションを探しています。EC2 インスタンスで SQL Server を実行している場合は、データのバックアップを作成し、デー

タを復元するための複数のオプションがあります。Amazon 上の SQL Server のデータを保護するためのバックアップ戦略EC2は次のとおりです。

- Windows Volume Shadow Copy Service (VSS) 対応の Amazon Elastic Block Store (Amazon EBS) スナップショットまたは を使用したサーバーレベルのバックアップ AWS Backup
- SQL Server でネイティブバックアップと復元を使用した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 には、次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 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 NetApp ONTAP
- 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 AWS Storage Gateway
- Amazon S3 for SQL Server 2022 への直接バックアップ

このセクションでは、以下を行います。

- ストレージスペースを節約するのに役立つ機能を強調表示
- 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します
- 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します

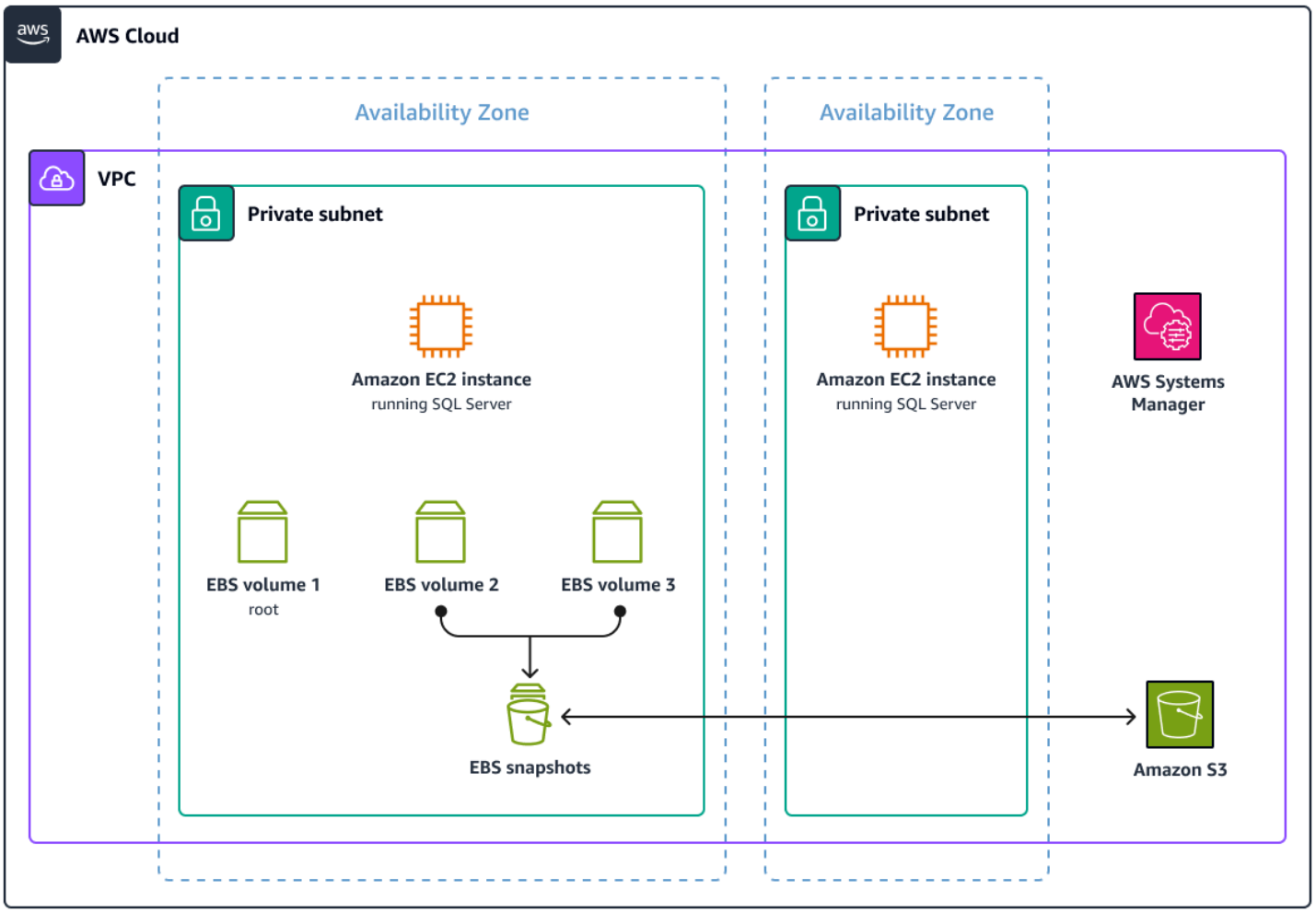
VSSが有効なスナップショットを使用したサーバーレベルのバックアップ

VSS対応スナップショットアーキテクチャでは、 AWS Systems Manager Run Command を使用してSQL、サーバーインスタンスに VSS エージェントをインストールします。Run Command を使用して、オペレーティングシステムとアプリケーション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ボリュームのスナップショットを作成し point-in-time、I/O を再開するワークフロー全体を呼び出すこともできます。

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プしてSQLサーバーファイル用の単一のファイルシステムを作成する場合、Amazon は 1 つのAPIコマンドを使用してクラッシュ整合性のあるマルチボリュームスナップショットEBSもサポートします。アプリケーション整合性VSSのある 対応EBSスナップショットの詳細については、Amazon ドキュメ

ントの「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」を参照してください。 VSS
EC2

次の図は、 VSSが有効な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が有効なスナップショットを使用する利点を以下から考慮してください。

- 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リカバリを提供します point-in-time。
- スナップショット から新しいSQLサーバーEC2インスタンスに復元できます。

- インスタンスが Amazon を使用して暗号化されている場合、EBSまたはデータベースが を使用してインスタンスで暗号化されている場合TDE、そのインスタンスまたはデータベースは同じ暗号化で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。
- スナップショットからEBSボリュームを復元すると、アプリケーションがそのボリュームにアクセスできるようになります。つまり、1 つ以上の基盤となるEBSボリュームをスナップショットから復元した後、すぐに SQL Server を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。つまり、スナップショットからEBSボリュームを復元した後、パフォーマンスに遅延が生じる可能性があります。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、高速スナップショット復元 (FSR) スナップショットを使用することで、この遅延を回避できます。
- 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。

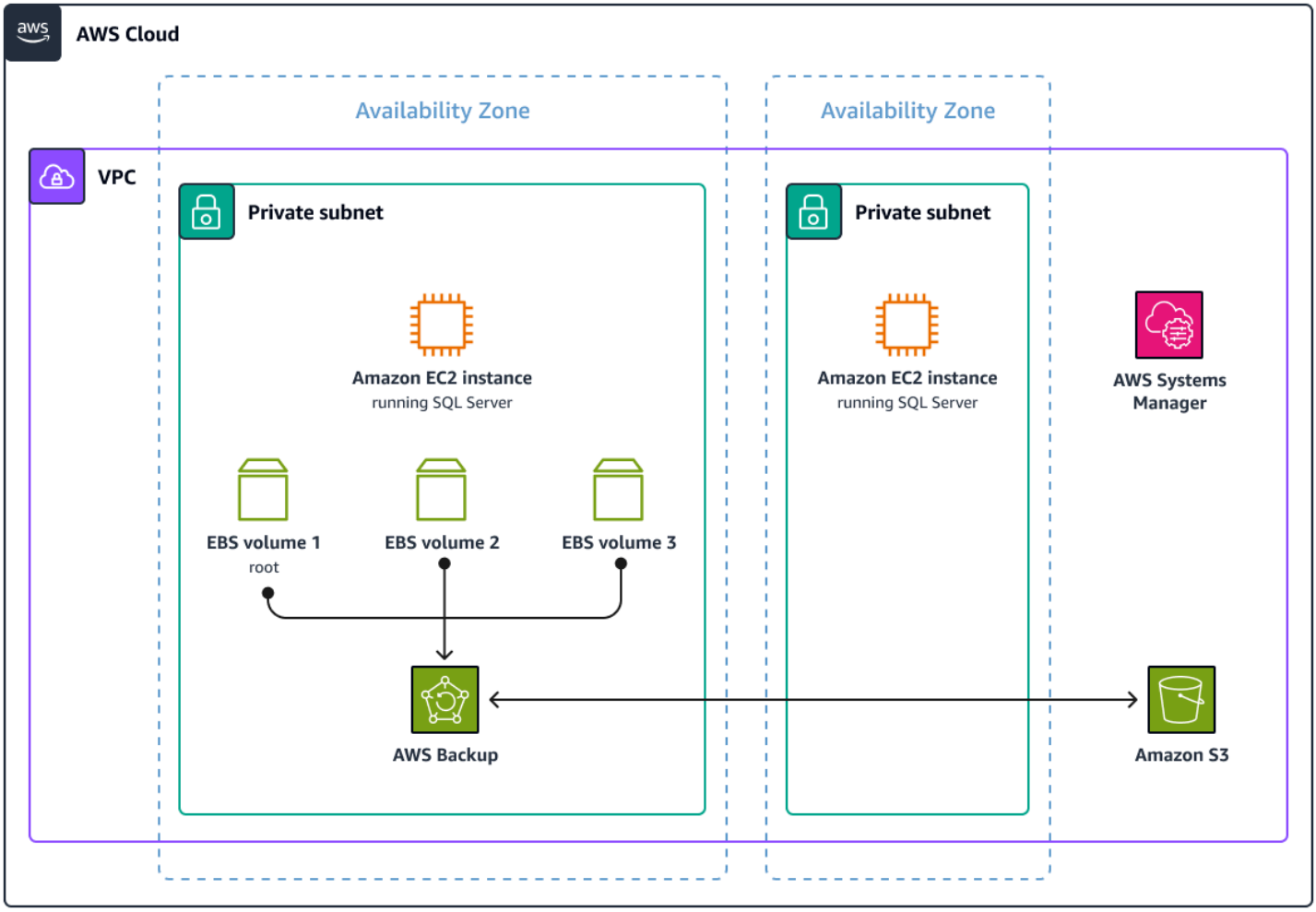
VSSが有効なスナップショットを使用する際には、以下の制限を考慮してください。

- SQL サーバー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 point-in-time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取得されるため、個々のデータベースを復元することはできません。
- インスタンス自体を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。

SQL を使用したサーバーバックアップ AWS Backup

AWS Backup を使用して、 全体のデータ保護を一元化および自動化できます AWS のサービス。 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 AWS Backup 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ためにも役立ちます。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、組織と AWS アカウント 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。

次の図は、 EC2を使用して 上の SQL Server のバックアップおよび復元ソリューションのアーキテクチャを示しています AWS Backup。



を使用して SQL Server をバックアップする利点を以下から検討してください AWS Backup。

- 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。
- バックアップ戦略は、複数の アカウントと にまたがって、組織全体で一元化できます AWS リージョン。
- 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を 全体で一元化できます AWS のサービス。
- 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。
- ソリューションは で動作しますTDE。

- 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。
- Server SQL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

これらのアプローチでは、ネイティブの Microsoft SQL Server バックアップ機能を使用します。Server SQLインスタンス上の個々のデータベースのバックアップを取得し、個々のデータベースを復元できます。

ネイティブSQLサーバーバックアップと復元の各オプションは、以下もサポートしています。

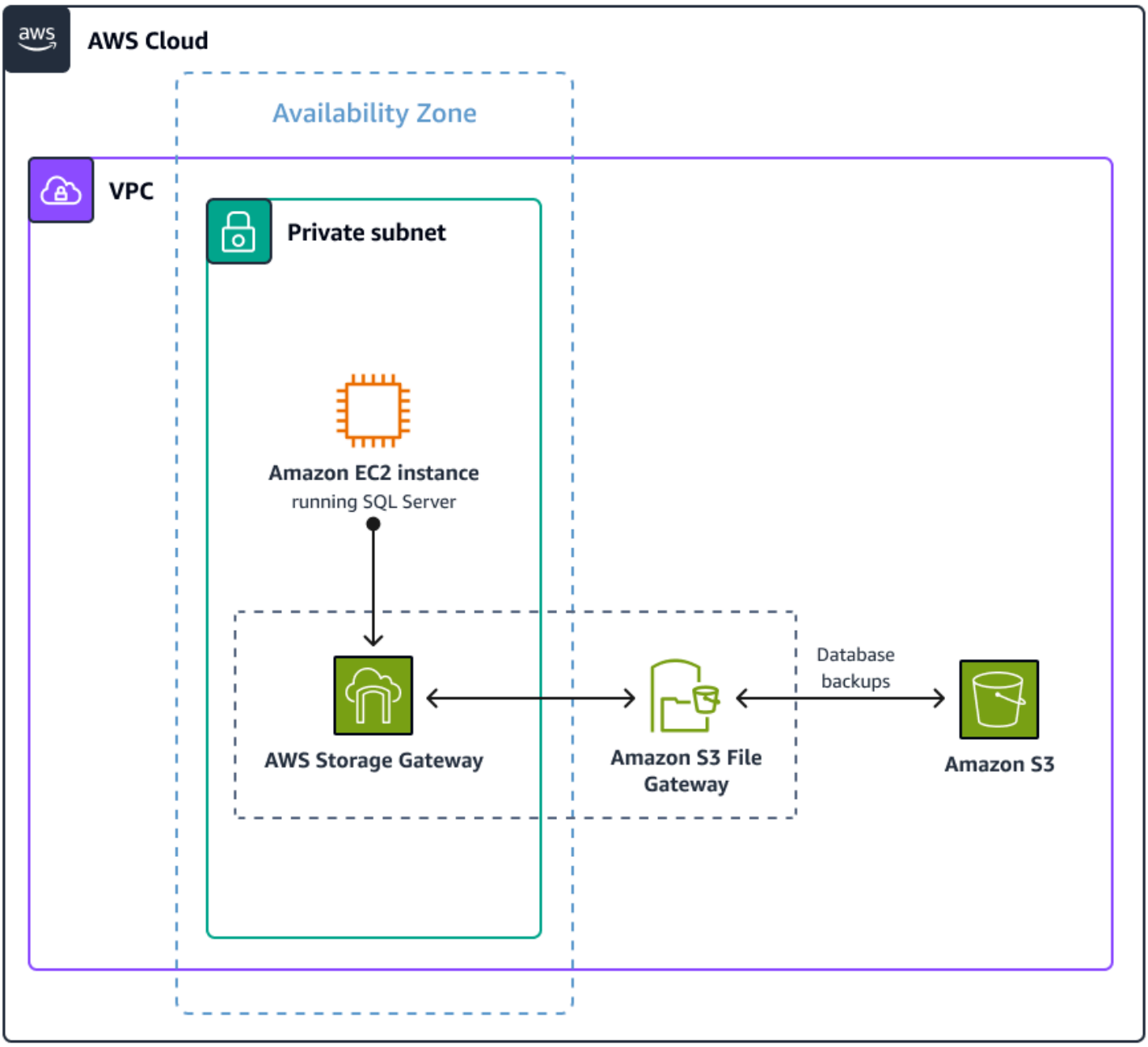
- 圧縮と複数ファイルのバックアップ
- フルバックアップ、差分バックアップ、T ログバックアップ
- TDE暗号化されたデータベース

SQL サーバーネイティブバックアップと Amazon S3 への復元

SQL Amazon 上のサーバーは、SQLサーバーデータベースのネイティブバックアップと復元EC2をサポートしています。Server SQL データベースのバックアップを取り、バックアップファイルを既存のデータベース、または新しい SQL Server EC2インスタンス、Amazon RDS for SQL Server、またはオンプレミスサーバーに復元できます。

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できるため、オンプレミスのストレージフットプリントが削減され、Amazon S3 を使用して耐久性、スケーラビリティ、コスト効率に優れたストレージを実現できます。

次の図は、Storage Gateway と Amazon S3 を使用するネイティブバックアップおよび復元ソリューションのアーキテクチャを示しています。



Storage Gateway でネイティブSQLサーバーバックアップを使用する利点を次に示します。

- ストレージゲートウェイをEC2インスタンス上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、または Storage Gateway ファイルキャッシュを介して送信されます。
- 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には、次の制限事項を考慮してください。

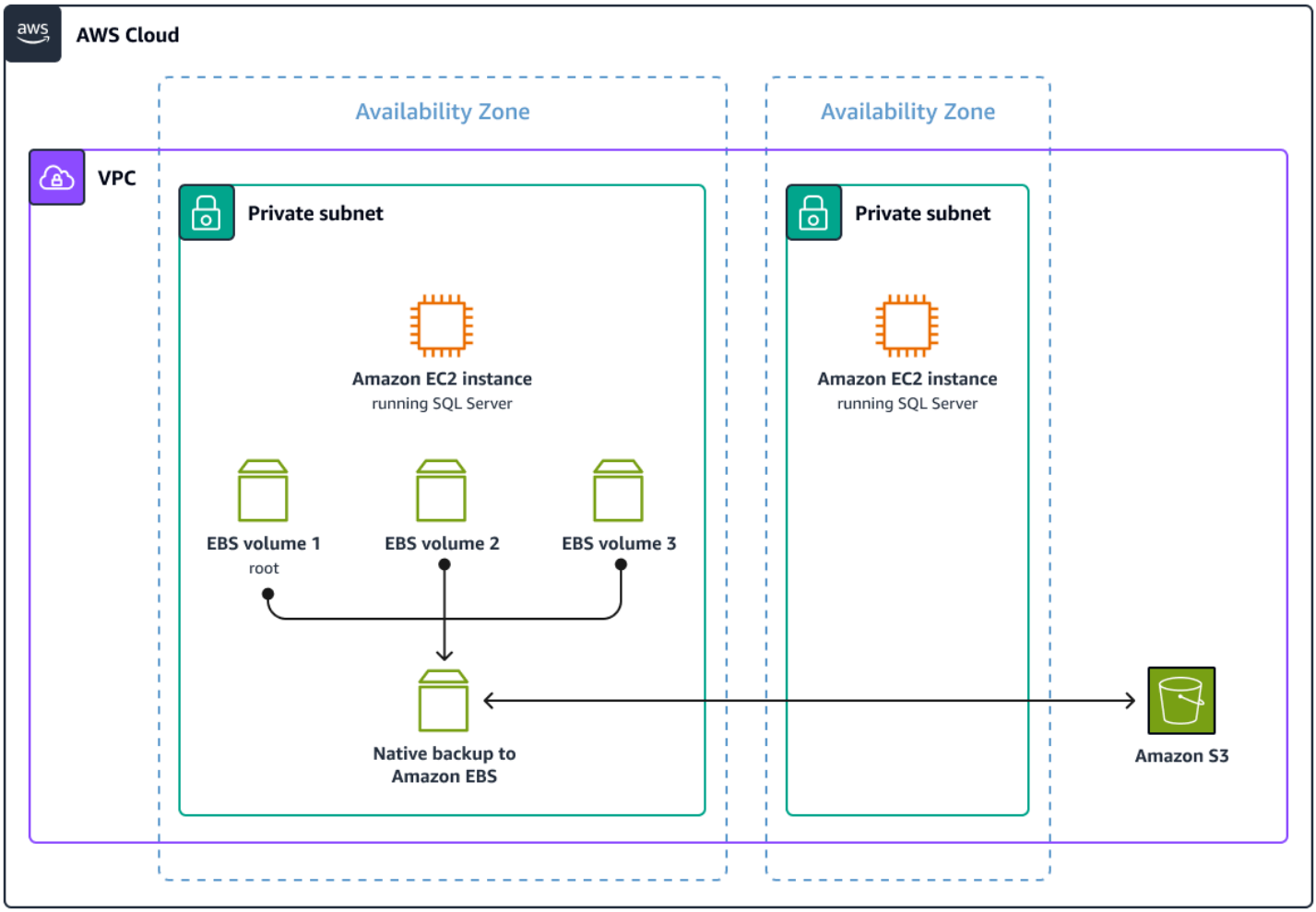
- バックアップと復元をデータベースごとに設定する必要があります。
- バックアップファイル用の Amazon S3 ライフサイクルポリシーの管理が必要になります。

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

SQL EBSボリュームへのサーバーネイティブバックアップ

Server SQL データベースのネイティブバックアップを取得し、Amazon EBSボリュームにファイルを保存できます。Amazon EBSは、パフォーマンスの高い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。これらはデタッチしてEC2インスタンスにアタッチできます。Server SQLは、同じEBSボリュームタイプまたは別のEBSボリュームタイプのEC2インスタンスでバックアップできます。別のEBSボリュームにバックアップする利点の 1 つは、コスト削減です。

次の図は、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ています。



Server SQLネイティブバックアップをEBSボリュームに使用する利点を以下から検討してください。

- 完全なEC2インスタンスを復元するのではなく、SQLサーバーインスタンス上の個々のデータベースのバックアップを取得し、個々のデータベースを復元できます。
- 複数ファイルのバックアップがサポートされています。
- Server Agent と SQL Server ジョブエンジンを使用して、バックアップSQLジョブを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。

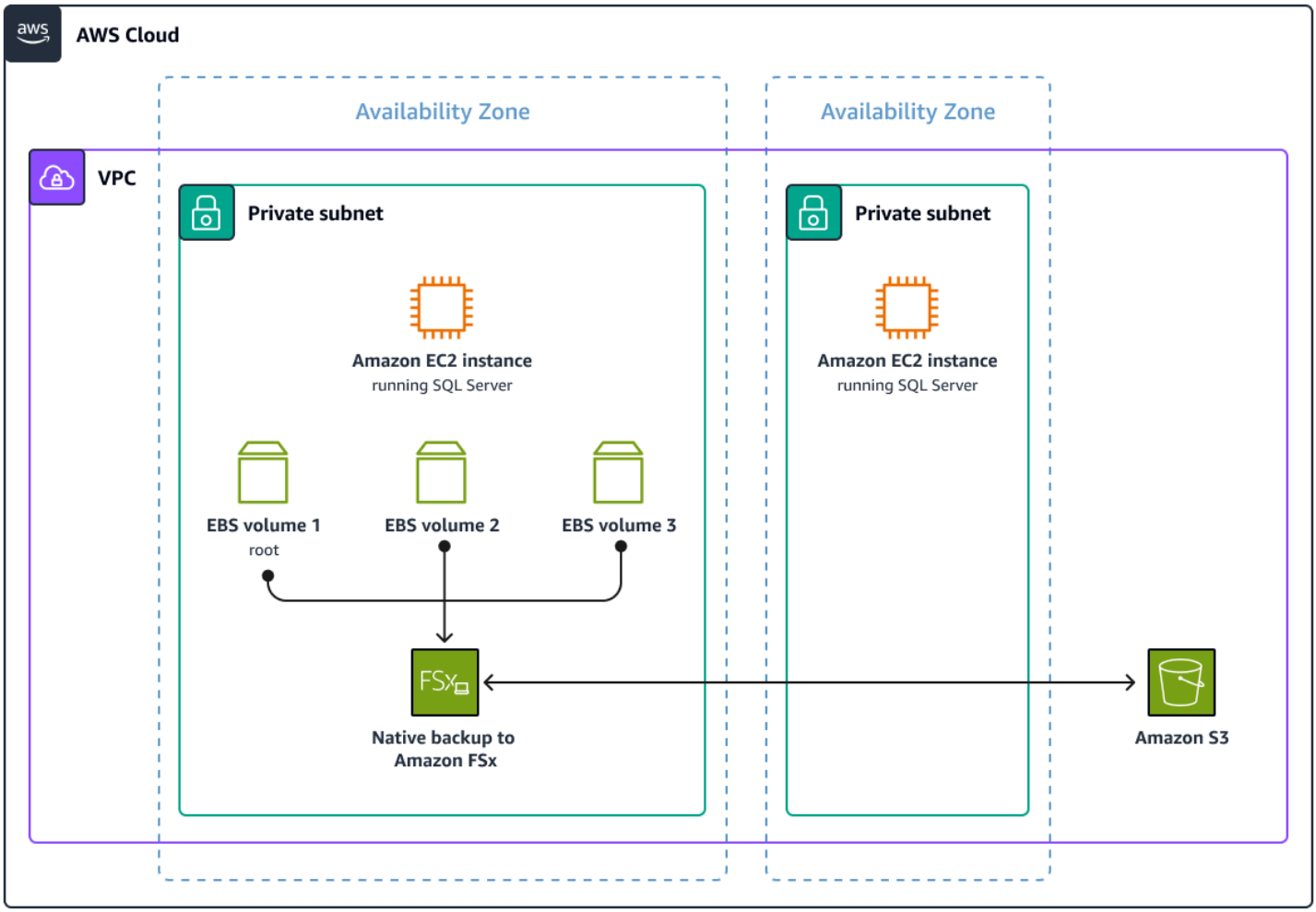
- EBS ボリュームから Amazon S3 にバックアップを手動で移動する必要があります。

- 大規模なバックアップの場合は、Amazon のディスク容量を管理する必要がありますEC2。
- EC2 インスタンスでは、Amazon EBS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 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ですEBS。

SQL Amazon FSx for Windows File Server へのサーバーネイティブバックアップ

Amazon FSx for Windows File Server は、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。 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 が導入されました。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に高可用性と冗長性 AWS を備えた Windows ファイルストレージを に簡単にデプロイできます。 では、SMB継続的可用性 (CA) ファイル共有 のサポート AWS も導入されました。Windows File Server FSxでは、 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。

次の図は、 for Windows File Server FSx へのネイティブSQLサーバーバックアップのアーキテクチャを示しています。



for Windows File Server で FSx にネイティブSQLサーバーバックアップを使用する利点を次に示します。

- Server SQL データベースを Amazon FSx ファイル共有にバックアップできます。
- 完全なインスタンスを復元するのではなく、SQLサーバーインスタンス上の個々のデータベースのバックアップを取り、個々のデータベースを復元できます。
- マルチパートバックアップがサポートされています。
- Server SQL Agent とジョブエンジンを使用して、バックアップジョブをスケジュールできます。
- インスタンスのネットワーク帯域幅は、Amazon に比べて高くなりますEBS。

for Windows File Server で FSx へのネイティブSQLサーバーバックアップを使用する際には、以下の制限を考慮してください。

- AWS Backup または FSxを使用して、Amazon 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。
- 大規模なバックアップでは、Amazon でのディスク領域管理に追加のオーバーヘッドが必要になる場合がありますFSx。
- EC2 インスタンスネットワークスループットはボトルネックになる可能性があります。
- for Windows File Server FSx 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です。

SQL FSxの Amazon へのサーバーバックアップ NetApp ONTAP

FSx for のスナップショットONTAPは常にクラッシュ整合性がありますが、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。 (SQLサーバーを含む特定のアプリケーションのプラグインを備えた NetApp SnapCenter オーケストレーションツール) を FSx で使用ONTAP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、追加コストなしでデータベースを保護、レプリケート、クローンできます。

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。 は、スナップショットをバックアップとして SnapCenter 参照します。このガイドでは、同じ命名規則を採用しています。 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 1 つのウィンドウ SnapCenter を提供します。特定のデータベースアプリケーションの 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 サーバーの 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以下の機能があります。

- フルバックアップとログバックアップのきめ細かなバックアップと復元のオプション
- インプレース復元と代替場所への復元

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

バックアップのコスト最適化

以下のオプションは、SQLサーバーバックアップを に保存する際のコストを削減するのに役立ちます AWS。

- バックアップファイルの作成中にSQLサーバー圧縮を有効にし、可能な限り小さなファイルをストレージに送信します。例えば、圧縮率が 3:1 の場合、ディスク容量を約 66% 節約していることを示します。これらの列で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 バケットへのバックアップでは、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。
- Windows File Server または FSxの FSxにバックアップする場合はONTAP、1 つのアベイラビリティーゾーンを使用して 50% のコスト削減 (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) を行います。料金については、「Amazon FSx for Windows File Server の料金」および「Amazon FSx for NetApp ONTAP Pricing」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、追加コストを削減できます。

バックアップのテスト結果をベンチマークする

このセクションでは、この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以下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 デベロッパーエディションを使用した r5d.8xlarge
- データベース仕様 – 1 TB サイズ、TDE無効

テストは、ソースとして r5d.8xlarge インスタンスと 1 TB SQL Server データベースを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従って設定され、ソースデータベースには 4 つのデータファイル (それぞれ 250 GB) と 1 つのログファイル (50 GB) が別々の gp3 ボリュームに分散していました。SQL Server ネイティブBACKUPコマンドには、10 個のバックアップファイルに書き込むこと、圧縮を使用し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量を減らすことが含まれます。すべてのテストケースで、ストレージ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	実行期間 (Appx)	バックアップレート	1 か月USDあたりのコスト*
ローカル EBS st1 へのネイティブバックアップ HDD、2 TB	データベース	00:30:46 分	554.7 Mbps	92.16 ドル
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 ドル
Windows File Server FSx用へのネイティブバックアップHDD、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 ドル
Windows File Server FSx のへのネイティブバックアップSSD、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 ドル
2 TB gp3 を使用した S3 File Gateway m6i.4xlarge (16 v CPU、64 GB) へのネイティブバックアップ	データベース	00:23:20 分	731.5 Mbps	470.42 ドル

バックアップオプション	レベル	実行期間 (Appx)	バックアップレート	1 か月USDあたりのコスト*
EBS VSS スナップショット	EBS ボリューム	00:00:02 秒 00:00:53 秒	N/A スナップショット	51 ドル
AWS Backup （AMI バックアップ）	AMI	00:00:04 秒 00:08:00 分	N/A スナップショット	$75
Amazon S3 へのネイティブ SQLサーバーの直接バックアップ (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 の へのネイティブバックアップ ONTAP （を使用 SnapCenter）	データベース	–	–	$440.20

上記の表では、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ています。
- 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成長率は 10% で、複数のバックアップでは 1 か月全体の変化率が 10% です。

テスト結果は、最速オプションが Windows File Server FSx 用の へのネイティブSQLサーバー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、パフォーマンスは遅く

なります。サーバーレベルのバックアップ (AMI) では、パフォーマンス、コスト、管理性を最適化 AWS Backup するために を使用することをお勧めします。

コスト最適化に関する推奨事項

Amazon で SQL Server をバックアップするための考えられるソリューションを理解することEC2 は、データを保護し、バックアップのニーズを確実に満たし、重要なイベントから復旧するための計画を立てる上で重要です。このセクションで調べたSQLサーバー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。

- 圧縮
- 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
- へのバックアップ URL

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。これらの推奨事項を組織に実装する場合は、アカウントチームに問い合わせ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できます。optimize-microsoft@amazon.com に E メールを送信して に連絡することもできます。

要約すると、以下をお勧めします。

- 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、Amazon S3 でバックアップされた Storage Gateway へのバックアップを最もコスト効率の高いオプションとして検討してください。

圧縮

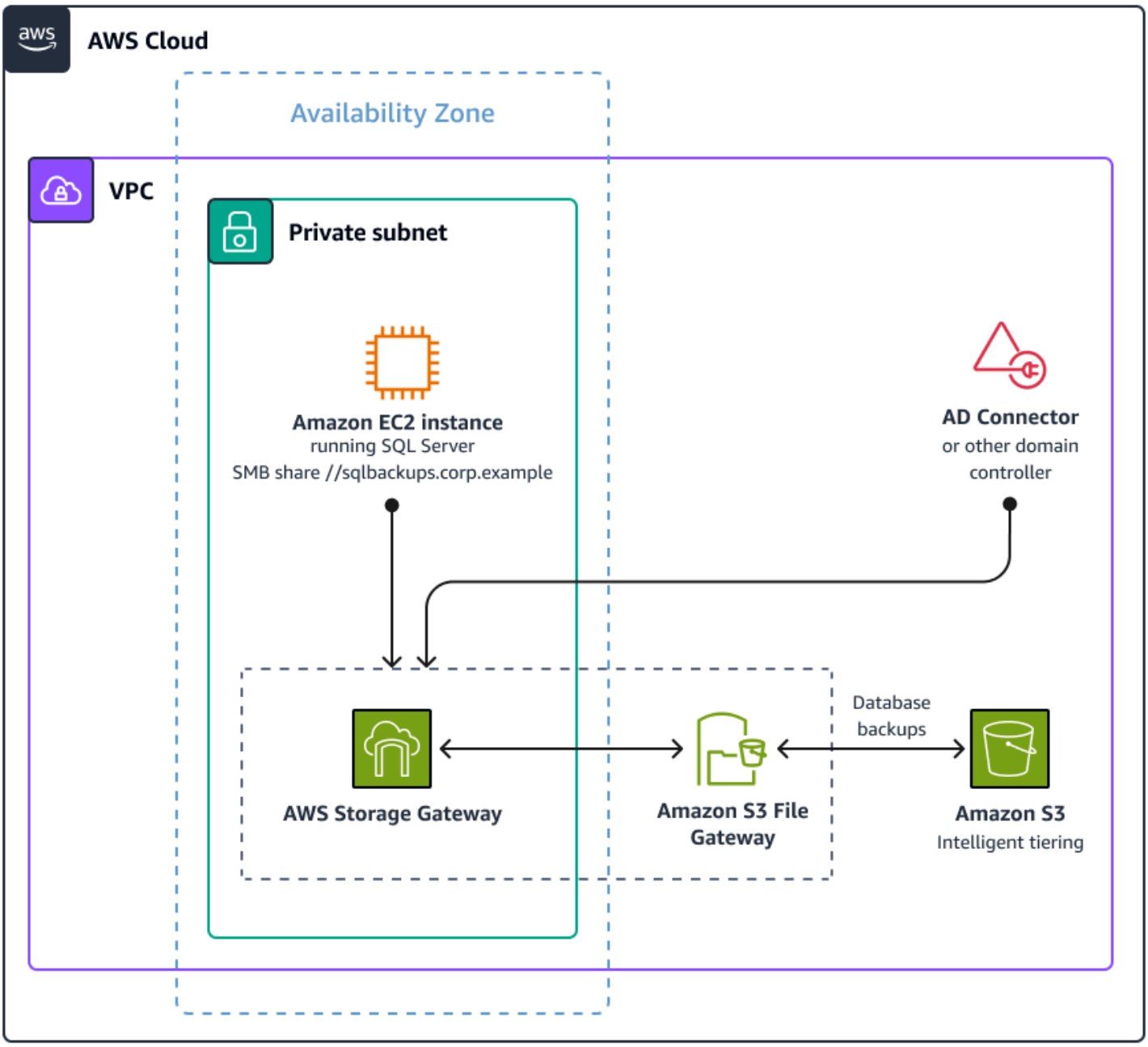
圧縮の目的は、各バックアップで消費されるストレージを減らすことです。これは、さまざまなストレージオプションにとって有益です。SQL サーバーバックアップの圧縮は、SQLサーバーインスタンス のレベルで有効にする必要があります。次の例は、バックアップデータベースで圧縮キーワードを追加する方法を示しています。

```
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では、Amazon S3 File Gateway ストレージクラス として Amazon S3 Intelligent-Tiering を有効にできます。 Amazon S3 https://docs.aws.amazon.com/filegateway/latest/files3/storage-classes.html#ia-file-gateway 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 File Gateway 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 に焦点を当てた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されていない可能性の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 Active Directory との通信を容易にできます。ゲートウェイで Active Directory 設定を構成するには、Active Directory へのリクエストをプロキシするドメインコントローラーIPsの Active Directory Connector 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。を使用して S3 Intelligent-Tiering AWS Management Consoleを有効にすることもできます。詳細については、 ドキュメントの AWS Amazon S3 Intelligent-Tiering の使用開始」を参照してください。

単一のアベイラビリティーゾーン

シングルアベイラビリティーゾーン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 Single-AZ オプションを選択します。Amazon は FSx、Windows Volume Shadow Copy Service 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、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、SMB ファイルシステムがプロビジョニングされてい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、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利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「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。

へのバックアップ URL

SQL Server 2022 の場合、 機能のバックアップURLにより 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 のすべての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 コストを削除する AWS ときに、 で実行される 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でクロスリージョンデータ出力コスト 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、ブログの Microsoft Workloads の「Amazon Amazon S3への Backup SQL Server データベース」の AWS 投稿を参照してください。

追加リソース

- Amazon 上の SQL Server のバックアップと復元のオプション EC2 (AWS 規範ガイダンス)
- Point-in-time RDS による Amazon の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 (ストレージブログ) NetApp SnapCenter で を使用してSQLサーバーワークロードを保護する NetApp ONTAPAWS
- Amazon S3 Intelligent-Tiering の使用を開始する (AWS Getting Started Resource Center)
- 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 に移行する EC2 (AWS 規範ガイダンス)

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

SQL サーバー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道のりを歩んでいる場合は、SQLServer などの商用データベースで課題に直面している可能性があります。商用データベースは高価で、顧客をロックインし、懲罰的なライセンス条項を提供します。このセクションでは、SQLサーバー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と、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。

Server SQL 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に統合します。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 サーバーレス 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 90% まで削減できます。さらに、 AWS では、Babelfish for Aurora Postgre SQL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、 でのSQL サーバー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。

データベースの提供

Windows 上の SQL Server から Amazon Aurora、Amazon RDS for My 、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SQL、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。以下の点を考慮します。

- Amazon の SQL Server Enterprise Edition から Amazon RDS for PostgreSQL または Amazon RDS for MySQL EC2に切り替えると、最大 80% のコスト削減につながります。
- Amazon の SQL Server Enterprise Edition から EC2 Amazon Aurora Postgre SQL互換エディションまたは Amazon Aurora My SQL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 アドレスの要件と、リレーショナルデータベースの費用対効果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と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です。ただし、他の商用ベンダーの同様の機能と比較して、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、移植性にはさらに利点があります。最適なオプションが Amazon RDS for Postgre SQL、Amazon RDS for My 、または Aurora のいずれであるかは、ビジネス要件を理解しSQL、必要な機能を最適なオプションにマッピングすることにあります。

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。

カテゴリ	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	優れたパフォーマンス	3 倍以上のパフォーマンス
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 つのリードレプリカ 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
ストレージ	最大 64 TB	最大 128 TB
ストレージ HA	1 つまたは 2 つのスタンバイを持つマルチ AZ、それぞれデータベースコピー	デフォルトでは、3 つのアベイラビリティーゾーンに 6 つのデータコピー
バックアップ	毎日の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	該当なし	100 GB データベースの高速クローン作成

カテゴリ	Amazon RDS for PostgreSQL または Amazon RDS for MySQL	Aurora PostgreSQL または Aurora MySQL
	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	

*大規模なトランザクションではフェイルオーバー時間が長くなる可能性があります

**Aurora Postgre で利用可能SQL

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USDあたりのコスト*	AWS Pricing Calculator (が必要 AWS アカウント)
Amazon RDS for SQL Server Enterprise エディション	3,750 ドル	見積り
Amazon RDS for SQL Server Standard エディション	2,318 ドル	見積り
SQL Amazon での Server Enterprise Edition EC2	2,835 ドル	見積り

データベースサービス	1 か月USDあたりのコスト*	AWS Pricing Calculator (が必要 AWS アカウント)
SQL Amazon の Server Standard Edition EC2	1,345 ドル	見積り
Amazon RDS for PostgreSQL	$742	見積り
Amazon RDS for MySQL	$712	見積り
Aurora PostgreSQL	1,032 ドル	見積り
Aurora MySQL	1,031 ドル	見積り

* ストレージ料金はインスタンス料金に含まれています。コストは`us-east-1`リージョンに基づきます。スループットと IOPSは前提条件です。計算は r6i.2xlarge インスタンスと r6g.2xlarge インスタンス用です。

コスト最適化に関する推奨事項

異種データベース移行では、通常、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、データをソースからターゲットデータベースに移行する必要があります。移行の最初のステップは、SQLサーバー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 Postgre 、Aurora My SQL、Postgre 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SQLおよび評価できますSQL。Babelfish for Aurora Postgre 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SQL。これにより、 AWS SCT と Compass は、移行戦略を決定する前に、関連する先行作業を理解するための強力なツールになります。続行すると、 はスキーマに必要な変更 AWS SCT を自動化します。Babelfish Compass の背後にある基本理念は、SQLデータベースを Aurora に移動させ、変更を加えない、またはごくわずかにすることです。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サーバーから Aurora へのデータの移行に労力が費やす前に、結果がわかります。

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 を使用して、スキーマの変更がない、または最小限

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、移行が加速する可能性があります。

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、以下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 コンパス
- 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、 ドキュメントの AWS SCT 「 SQLを使用したアプリケーションの変換 AWS SCT」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にポイントし、評価を実行します。次に、 はスキーマAWS SCT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、複雑さと移行の労力、適切な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。を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よって AWS SCT 生成されたエグゼクティブサマリーの例を示しています。

ターゲットプラットフォーム	自動または最小限の変更	複雑なアクション

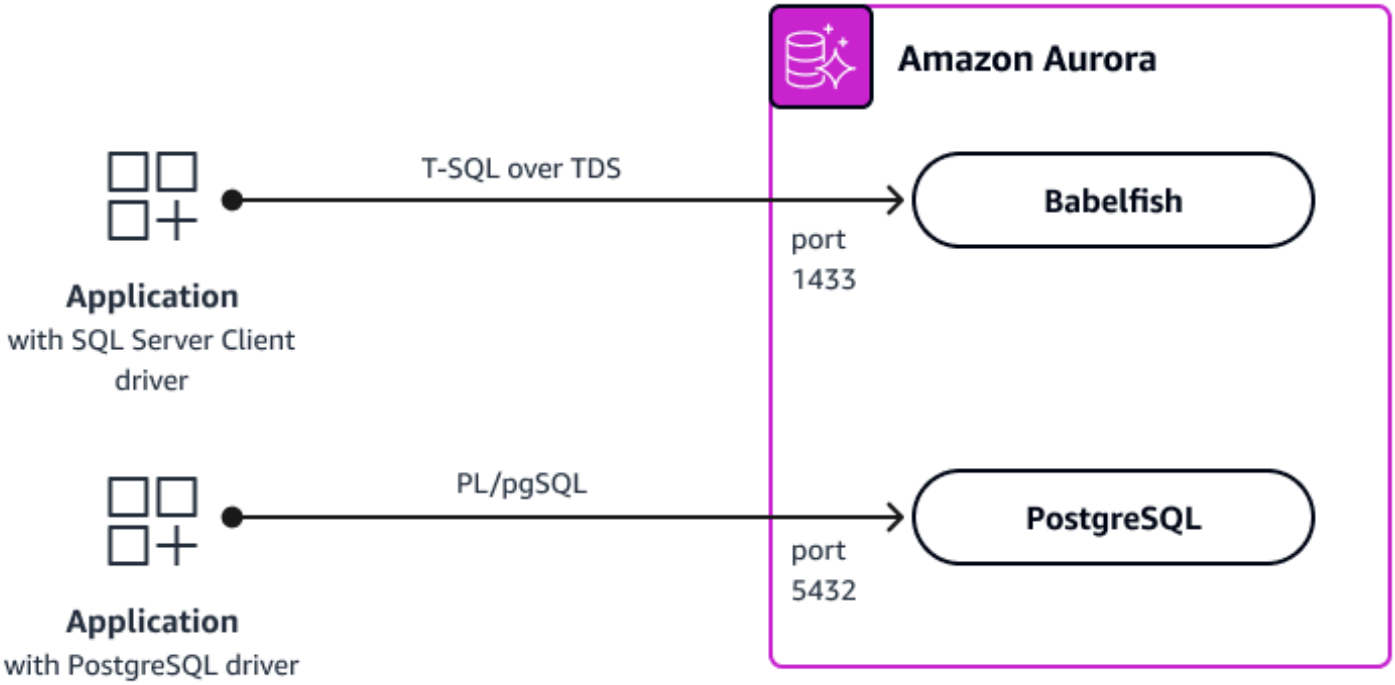
	ストレージオブジェクト	コードオブジェクト	変換アクション	ストレージオブジェクト		コードオブジェクト	
Amazon RDS for MySQL	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56
Amazon Aurora My SQL 互換エディション	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56
Amazon RDS for PostgreSQL	60 (98%)	12 (52%)	54	1 (2%)	1	11 (48%)	26
Amazon Aurora Postgre SQL互換エディション	60 (98%)	12 (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サーバークライアントからのデータベース接続を受け入れる機能を備えた Aurora PostgreSQL を拡張します。Babelfish を使用すると、SQLサーバー用に構築されたアプリケーションは、コードの変更が少なくSQL、データベースドライバーを変更することなく、Aurora Postgre と直接連携できます。Babelfish は Aurora PostgreSQL をバイリンガルに変換して、Aurora PostgreSQL が T 言語SQLと PL/pgSQL 言語の両方を操作できるようにします。Babelfish は、SQLサーバーから Aurora Postgre に移行する労力を最小限に抑えますSQL。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。引き続き T SQLポスト移行を使用できますが、PostgreSQL ネイティブツールを開発に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 Postgre データベースに直接および同時に接続できますSQL。



Babelfish は、特定のSQLサーバー T SQL機能をサポートしていません。このため、Amazon はSQL ステートメントの分析を行い line-by-line、Babelfish がサポートしていないステートメント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 評価には 2 つのオプションがあります。 は、SQLサーバーデータベースと 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。これは、Aurora Postgre 用の Babelfish の新しいリリースに合わせて Compass ツールが更新されるため、推奨されるソリューションですSQL。

Babelfish コンパス

Babelfish Compass は、Babelfish for Aurora Postgre の最新リリースと一致する無料のダウンロード可能なツールですSQL。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サーバーデータベーススキーマに対して実行されます。Server Management Studio () などのツールを使用して、ソースSQLサーバーSQLデータベーススキーマを抽出することもできますSSMS。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サーバースキーマと Babelfish の互換性、および移行前に変更が必要かどうかの詳細を示す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することもできます。

評価と変更が完了したら、 SSMSや sqlcmd などのSQLサーバー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「Babelfish 投稿を使用したSQLサーバーから Amazon Aurora への移行」を参照してください。

AWS Database Migration Service

スキーマが移行されたら、 AWS Database Migration Service （ AWS DMS) を使用してダウンタイムを最小限に抑え AWS てデータを 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ると、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カットオーバーアクティビティを実行できます。 AWS DMS 現在、Aurora PostgreSQL ターゲットに対して Babelfish で完全な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 AWS Database Migration Serviceを使用する」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サーバーデータベースを Amazon RDS for SQL Server に移行する AWS DMSを参照してください。

追加リソース

- Microsoft SQL Server、Hello Babelfish (AWS ニュースブログ)
- (データベースブログ) SQLを使用してデータベーススキーマとアプリケーションを変換する AWS Schema Conversion Tool CLIAWS
- 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、サーバーを 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 Postgre への移行後のデータベースオブジェクトの検証SQL (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 用の 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。

にSQLサーバー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サーバーワークロードを実行する AWS お客様は、通常、Amazon EBS、FSx、Amazon NVMe、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、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、およびバックアップファイルを保存および処理するために使用されるネットワーク接続ストレージです。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 と gp3) とプロビジョンド IOPS SSD (io1、io2、io2BX) が含まれます。以下の点を考慮します。

- r5d などの一部のEC2インスタンスでは、ローカルがホストインスタンスにNVMeSSDs物理的にアタッチされています。これらのボリュームは、SQLサーバーtempdb またはバッファプール拡張に一般的に使用されるブロックレベルの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共有ストレージです。Amazon FSxは、高可用性のSQLサーバーフェイルオーバークラスターインスタンス (FCI) 設定でSQLサーバー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サーバーデータとログファイルをホストするため、EC2インスタンスEBSの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 サーバーネイティブバックアップファイル、AMIs、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。

SSD Amazon のストレージタイプ、パフォーマンス、コスト EBS

SSD Amazon のストレージコストは、耐久性とパフォーマンスの向上に伴ってEBS一般的に増加します。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-backed 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「ソリッドステートドライブ (SSD) ボリューム」セクションの表を参照してください。

Amazon を使用して、SSDパフォーマンス CloudWatch のモニタリング、傾向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。でSQLサーバーワークロードを実行している場合は AWS、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、使用容量と空き容量など、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ために、詳細なモニタリングとCloudWatch カスタムメトリクスのデプロイを有効に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ストレージとプロビジョニング過剰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。

SSD Amazon のストレージコストEBSも、割り当てられた容量によって異なります。以下の表は、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は 1 TB の容量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 ドル	
gp3	3,000	250	86.92 ドル	15%

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
io1	16,000	500	1,168 ドル	
io2	16,000	500	1,168 ドル	
gp3	16,000	500	146.92 ドル	87%
io2bx	16,000	4,000	1,168 ドル	
gp3	16,000	1,000	181.92 ドル	84%

> **Note**
>
> 前の表のパフォーマンスとコストのメトリクスは、 からの見積りに基づくボリュームあたりのものです AWS Pricing Calculator。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 Pricing Calculator。

Amazon gp3 EBS SSD 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 および 500 スループット未満のワークロードで 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% IOPS を節約できます MiBps 。

io2 Block Express (io2BX) ボリュームは、通常の io2 ボリュームよりもパフォーマンスが向上します。16,000 ではIOPS、io1 または io2 ボリュームは 500 MiBps スループットしか実行できず、io2BX ボリュームは最大 4,000 MiBps スループットまで設定できます。io1 ボリュームと io2 ボリュームと比較して、io2BX ボリュームは 16,000 から 64,000 までのスループットの 4 倍以上を IOPS、まったく同じ価格で提供します。通常の io2 ボリュームは、io2BX io2BX-supported ボリュームに変換できます。 EC2io2BX-supportedEC2インスタンスのリストについては、Amazon EBSドキュメントの「プロビジョニングされたIOPSSSD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、 を使用して毎月のコストAWS Pricing Calculatorを見積もり、耐久性、パフォーマンス、容量のトレードオフに基づいてコストへの影響を理解できます。

Amazon の一般的なSSDコスト最適化 EBS

保存しているものを評価し、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サーバーバックアップに最適な優れた価格帯、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。Server の古いバージョンを実行している場合はSQL、Amazon EBSHDDボリュームにバックアップし、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のではなく、53% のコスト削減が可能です。

次の表は、Amazon gp3、Amazon st1、および Amazon S3 EBS 上の 1 TB EBS HDD 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	容量	料金 pm
EBS gp3 500 MiBps	1 TB	96.92 ドル
EBS st1 バースト 500 MiBps		$46.08
S3 Standard		23.55 ドル
S3 標準 (低頻度アクセス)		12.80 ドル
S3 Glacier Deep Archive		$1.03

> **Note**
>
> 前の表のコストメトリクスは、 の見積りに基づいています AWS Pricing Calculator。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 Pricing Calculator。

以下を考慮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 にアップグレードすると、耐久性とパフォーマンスの柔軟性が向上します。
- 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。

- 容量要件と高性能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、 mix-and-matchアプローチ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サーバーのインストール、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 ボリュームの容量とコストを削減できます。これは、高性能ユーザーデータベース専用です。
- でSQLサーバーデータベースをホストする場合は AWS、データベースごとに複数のSQLサーバー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容量の要件が軽減され、結果としてコストを削減できます。
- 実稼働ワークロードが io1 や io2/io2BX などの高性能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、非実稼働ワークロードの gp3 ボリュームを検討してください。
- ストレージの使用率を経時的に追跡してトレンド化し、使用量の急増や予期しないコストを簡単に特定できます。
- 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AWS Compute Optimizerに を使用します。
- の弾力性を使用して、Amazon のSSDボリュームのパフォーマンスと容量のニーズ AWS を調整しますEBS。オンプレミス環境とは異なり、将来のワークロードのためにストレージのパフォーマンスと容量を過剰にプロビジョニングする必要はありません。既存のSQLサーバーワークロードをに移行 AWS し、データベースをオンラインに保ちながら、必要に応じてパフォーマンスや容量を調整できます。

追加リソース

- Amazon EBSボリュームタイプ (Amazon EBSドキュメント)
- Amazon Elastic Block Store (Amazon EBS) (Amazon EBSドキュメント)
- プロビジョニングされたIOPSSSDボリューム (Amazon EBSドキュメント)
- SSD インスタンスストアボリューム (Amazon EC2ドキュメント)
- Amazon の Amazon CloudWatch メトリクス EBS (Amazon EBSドキュメント)
- Amazon EC2ストレージ最適化インスタンスの仕様 (Amazon EC2ドキュメント)
- Amazon FSx for (ストレージブログ) NetApp SnapCenter で を使用してSQLサーバーワークロードを保護する NetApp ONTAPAWS
- Amazon EC2 FAQ (AWS 製品ページ)

Compute Optimizer を使用してSQLサーバーライセンスを最適化する

を使用してSQLサーバー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) 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しますEC2。Compute Optimizer は、ライセンスコストを削減するための自動レコメンデーションを提供できます。Compute Optimizer 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、推奨される保存機会、EC2インスタンスオンデマンド料金、および時間単位の独自のライセンス (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るかどうかを決定するのに役立ちます。

Compute Optimizer は、推定ワークロードタイプEC2によって Amazon 上のSQLサーバーインスタンスを自動的に検出します。ライセンスのレコメンデーションを表示するには、Compute OptimizerでSQLサーバー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サーバーエンタープライズエディション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お勧めします。

Compute Optimizer を使用して、SQLサーバーワークロードを実行する Amazon EC2インスタンスのサイズ決定に関する推奨事項を作成することもできます。詳細については、このガイドの「Compute Optimizer を使用してSQLサーバーのサイズを最適化する」を参照してください。

コスト最適化に関する推奨事項

Compute Optimizer のライセンスレコメンデーションは、Microsoft SQL Server で使用している機能を評価し、ワークロードに最も費用対効果の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 よりも大幅に高価です。詳細については、このガイドのSQL「サーバーエディションの比較」および Microsoft ウェブサイトのSQL「サーバー 2022 の料金」を参照してください。SQL サーバーフリートを評価し、レコメンデーションを提供するように Compute Optimizer を設定する時間をかけると、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。

- テーブルを使用して、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) と Compute Optimizer のレコメンデーションを比較します。
- 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 ドキュメントの商用ソフトウェアライセンスのレコメンデーションの詳細を表示する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、「商用ソフトウェアライセンスのメトリクス」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てください。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
   - スタンドアロン用の Compute Optimizer へのアクセスを許可するポリシー AWS アカウント
   - 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 Application Insights 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のレコメンデーションを有効にするポリシー」を参照してください。
3. Microsoft SQL Server データベース認証情報を使用して CloudWatch Application Insights を有効にします。手順については、 CloudWatch ドキュメントの「モニタリング用のアプリケーションのセットアップ」を参照してください。

> **Note**
>
> 商用ソフトウェアライセンスのレコメンデーションを生成するには、少なくとも 30 時間のメトリクス CloudWatch データが必要です。詳細については、CloudWatch 「メトリクス要件」を参照してください。

4. 次のSQLクエリを使用して、 CloudWatch Application Insights の最小権限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、新しいサービスが有効になります PrometheusSqlExporterSQL。

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、 アカウントの「オプト」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 で、ナビゲーションペインでライセンスを選択します。
7. 結果列で、不十分なメトリクス結果が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か、アクセス許可が不十分であることが検出された場合、この検出結果を返します。詳細については、「理由の検出」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. ライセンスレコメンデーションを有効にする を選択します。
8. 結果列で、最適化されていない結果があるインスタンスを検索します。Compute Optimizer は、Amazon EC2インフラストラクチャが支払い対象の Microsoft SQL Server ライセンス機能を使用していないことを検出した場合に、この検出結果を返します。詳細については、「理由の検出」を参照してください。これらの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、「レコメンデーションを実装する」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SQL AWS Systems Manager サーバーエンタープライズエディションのダウングレード」を参照してコストを削減します (AWS ブログ) 。

追加リソース

- (AWS ブログ) を使用して Microsoft SQL Server のライセンスコストを削減 AWS Compute Optimizerする

- とは AWS Compute Optimizer (AWS ドキュメント)
- 商用ソフトウェアライセンスのレコメンデーションの表示 (AWS ドキュメント)
- Microsoft SQL Server エディションをダウングレードする (AWS ドキュメント)
- (AWS) 上の Microsoft SQL Server AWS
- (AWS) での Microsoft ライセンス AWS
- Microsoft SQL Server 2019 の料金 (Microsoft)
- Microsoft SQL Server 2022 の料金 (Microsoft)

Compute Optimizer を使用してSQLサーバーのサイズを最適化する

概要

AWS Compute Optimizer は、データベース管理者 (DBAs) が Amazon Elastic Compute Cloud (Amazon EC2) 上の 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、推定ワークロードタイプとして SQL Server のサポートが含まれています。推定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特定できます。

この機能を使用すると、SQLServer などのサポートされている推定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、オーバープロビジョニングされたSQLサーバー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のサイズを小さくすることで、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サーバーライセンスのレコメンデーションを行うこともできます。詳細については、このガイドの「Compute Optimizer を使用したSQLサーバーライセンスの最適化」を参照してください。

Compute Optimizer を設定する

SQL サーバー推論ワークロードで 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を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、オプトインすると、そのアカウントの Compute Optimizer のみが有

効になります。組織管理アカウントでは、そのアカウントでのみ Compute Optimizer を有効にするか、組織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。レコメンデーションを生成するには、過去 14 日間に少なくとも 30 時間の 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「メトリクス要件」と「拡張インフラストラクチャメトリクス」を参照してください。

Compute Optimizer は、v 、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CPU、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、レコメンデーションを行うために使用されるデータを分析できます。このデータを使用すると、SQLサーバー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を変更する方法の詳細については、Amazon EC2ドキュメントの「インスタンスタイプを変更する」を参照してください。

追加リソース

- 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 インスタンスレコメンデーションの表示 (AWS ドキュメント)

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

概要

AWS Trusted Advisorは、ベストプラクティスに従う AWS のに役立つ推奨事項を提供します。使用状況、設定、支出を分析することで、 は、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SQLサーバーワークロードを運用する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。

コスト最適化に関する推奨事項

Trusted Advisor は、Amazon Elastic Compute Cloud (Amazon) でSQLサーバー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EC2。チェックでは、SQLサーバー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改善できます。

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

- Microsoft SQL Server 用に過剰プロビジョニングされた Amazon EC2インスタンス — このチェックは、SQLServer 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 vCPU の制限を超えた場合に警告します。例えば、SQLServer Standard Edition のインスタンスでは、最大 48 を使用できますvCPUs。SQL Server Web を使用するインスタンスは、最大 32 を使用できますvCPUs。

エディション	vCPU 最小	vCPU 最大。
Web	4	32
標準	4	48
エンタープライズ	4	OS の制限

- Microsoft SQL Server の Amazon EC2インスタンスの統合 – このチェックは、Amazon EC2インスタンスを分析し、インスタンスのSQLサーバーライセンスの最小数に満たない場合に警告します。より小さなSQLサーバーインスタンスを統合することで、コストを削減できます。ライセンスに含まれる小さなSQLサーバーインスタンスが多数ある場合は、統合を検討してください。Microsoft SQL Server 2019 ライセンスガイド によると、SQLサーバーにはインスタンスごとに最低 4 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。決定は、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行うことができます。統合は、SQLサーバーのウェブ、スタンダード、エンタープライズエディションでサポートされています。詳細については、SQL「サーバーデータベースの統合」 (Microsoft ブログ記事) を参照してください。

AWS では、大規模な本番データベースを 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ものを統合することができます。これは現在のSQLサーバー使用状況によって異なります。使用頻度の低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。

Trusted Advisorを設定する

でSQLサーバー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 Management Console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レコメンデーション 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、Microsoft SQL Server の Amazon EC2インスタンス統合のステータスと、Microsoft SQL Server チェックの過剰プロビジョニングされた Amazon EC2インスタンス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. チェックの指示に従って、SQLサーバー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します。
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。

追加リソース

- Trusted Advisor チェックリファレンス (AWS ドキュメント)
- (AWS) 上の Microsoft SQL Server AWS
- (AWS) での Microsoft ライセンス AWS
- SQL Server 2019 の料金 (Microsoft)
- AWS Launch Wizard for SQL Server (AWS ドキュメント)

コンテナ

モダナイゼーションは、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駆動型にアプリケーションを再設計する、SQL Server から Amazon Aurora または専用の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モダナイゼーションパスは、他のモダナイゼーションオプションよりも労力がかかりません。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、デプロイなど、開発ライフサイクルの段階を簡単に自動化できます。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。
- 総所有コスト (TCO) の削減 — コンテナへの移行により、ライセンス管理ツールとエンドポイント保護ツールへの依存度も低下します。コンテナは一時的なコンピューティング単位であるため、パッチ適用、スケーリング、バックアップと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効率的です。これは、コンテナを使用して、分離性を向上させ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。
- リソース使用率の向上 – 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に比べて効率的です。これにより、分離性が向上し、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、開発チームにコンテナテクノロジーと DevOpsプラクティスのスキルを高めます。

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動する
- Amazon ECS での AWS Fargate タスクのコストの最適化
- 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、「Amazon Web Services and Microsoft: よくある質問」の「ライセンス」セクションを参照するか、microsoft@amazon.com にメールでお問い合わせください。 https://aws.amazon.com/windows/faq/#licensing-q

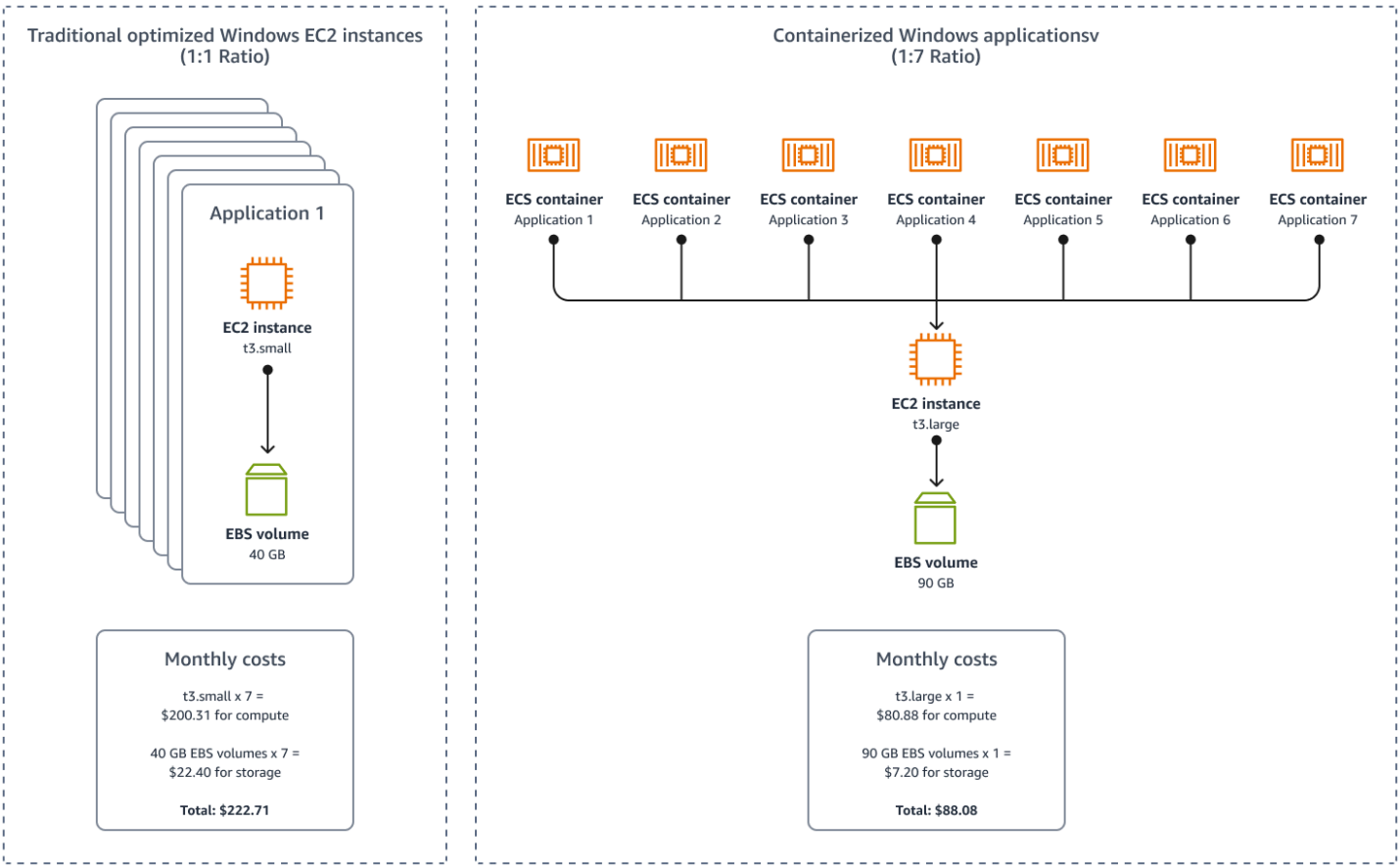
Windows アプリケーションをコンテナに移動する

概要

CNCF 年次調査 2021 によると、組織の 96% がコンテナを使用して、または評価してインフラストラクチャをモダナイズしています。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、アプリケーションを実行する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方法でコンテナを実行するための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 コンテナに移動することで達成できる追加の節約額を示しています。



AWS OLA は、企業が個々の 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で示されているように、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、企業はこれらの節約を実現できます。

Not sharable | Nearing end of life | Approximately 15% overhead | The OS accounts for 1.2 GB per instance | Optimized for EC2

	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

詳細な分析では、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の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します) 。このシナリオでは、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、予備の 3 GB の RAM を保持できます。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とストレージ全体で平均 64% のコスト削減を実現できます。

コスト最適化に関する推奨事項

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 の Windows フットプリントを削減する

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少数の Amazon EC2 インスタンスに統合できるため、Windows on Amazon EC2 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 用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) を使用するための 1:7 の比率 (EC2 インスタンスのタイプ/サイズによって大幅に増加する可能性がある) 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、Windows on Amazon EC2 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 インスタンスのライセンスを取得する場合、そのインスタンスで実行されているコンテナのライセンスは必要ありません。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、新しい Amazon Elastic Block Store (Amazon EBS) ボリュームを作成して支払い、オペレーティングシステムを格納します。これにより、コストはそれに合わせてスケールされます。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。

end-of-support サーバーをコンテナに移行する

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、リスクの軽減、運用効率、スピード、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必要がなくなります。

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 APIs (COM Interop など) を必要とする場合です。この場合、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動すると、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、将来的にオペレーティングシステムのアップグレードが簡単になります。

サードパーティーの管理ツールとライセンスを削除する

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一時的なものであり、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きめ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 を選択することで、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 の正確な量を定義できます。実際には、Windows コンテナのコンテナレベルの CPU とメモリを指定することをお勧めします。このレベルの粒度により、コスト上の利点が得られます。次のコード例を考えてみましょう。

```
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、デプロイなど、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。コンテナは一時的なコンピューティング単位であるため、パッチ適用、スケーリング、バックアップと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効率的です。

スキルギャップを埋める

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ています。これには、実践的なコンサルティングと有効化が含まれます。

.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動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 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 AWS。

ライセンスコストの削減

Windows の .NET Framework から Linux 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% のコスト削減になります。

最新の機能強化にアクセスする

Windows の .NET Framework から Linux 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 などの最新の機能強化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の価格が 40% 向上します。

セキュリティとパフォーマンスの向上

Windows の .NET Framework から Linux 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、コンテナを分離して、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

Internet Information Services (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という次の利点を考慮してください。

- セキュリティ — コンテナは、IIS レベルでの分離では実現されない、すぐに使えるレベルのセキュリティを提供します。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コントロールするよりも悪用するほど難しい脆弱性です。
- 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 インスタンスにまたがる複雑な分散メソッドも提供します。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、これらの利点は得られません。
- 管理オーバーヘッド — サーバーネーム表示 (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ない場合) 、エンドポイント保護など、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も対処する必要があります。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、時間がかかり、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なように設計されています。最終的に、Windows コンテナを代わりに使用すれば、デプロイはより高速で、より安全で、繰り返し可能です。

次のステップ

最新のインフラストラクチャに投資してレガシーワークロードを実行すると、組織に大きなメリットがもたらされます。 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、「」の「コンテナ AWS」を参照してください。

追加リソース

- ECS キャパシティープロバイダーと 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)
- Amazon ECS と のコスト最適化チェックリスト 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ケーションに関する考慮事項 (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 での AWS Fargate タスクのコストの最適化

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 タスクの任意のサイズで構築され、再検討されることはありません。これにより、Fargate タスクのオーバー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レコメンデーションを提供し、Fargate で実行されている Amazon Elastic Container Service (Amazon ECS) サービスのタスク CPU とメモリを最適化する方法を示します。Compute Optimizer は、これらのレコメンデーション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、節約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 のレコメンデーションは、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。アプリケーションの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、Fargate タスクがオーバーサイズになり、アイドル状態のリソースのコストが増加します。Compute Optimizer を使用すると、適切なサイジングの機会を積極的に表示できます。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動する」セクションを参照してください。

コスト最適化に関する推奨事項

このセクションでは、Compute Optimizer を使用して Fargate タスクで Amazon ECS を適切なサイズに設定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 の結果を消費する
- 適切なサイズにタスクをタグ付けする
- コスト配分タグによる AWS 請求ツールの使用を有効にする
- 適切なサイジングのレコメンデーションを実装する
- Cost Explorer でコストの前後に確認する

Compute Optimizer を有効にする

組織AWS Compute Optimizerレベルまたは の単一アカウントレベルで を有効にできます AWS Organizations。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、適切なサイジングをアクティビティではなく繰り返しのアクティビティに point-in-timeすることができます。

組織レベル

ほとんどの組織では、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、マルチアカウントおよびマルチリージョンで組織を可視化し、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。これを組織レベルで有効にするには、次の手順を実行します。

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を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、レコメンデーションを参照できます。

> **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、 にアクセスできない場合は AWS Organizations、そのアカウントとリージョンで 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、生成に最大 12 時間かかる場合があります。Compute Optimizer では、Fargate で Amazon ECS のレコメンデーションを生成するために、過去 14 日間に 24 時間のメトリクスが必要であ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「Fargate での Amazon ECS サービスの要件」を参照してください。

Compute Optimizer は、Fargate の Amazon ECS サービスの次の Amazon CloudWatch および Amazon ECS 使用率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る CPU 容量の割合。
- `MemoryUtilization` – サービスで使用されるメモリの割合。

Compute Optimizer の結果を消費する

1 つのアカウントと 1 つのリージョン内で適切なサイジング変更を行うことに焦点を当てた例を考えてみましょう。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。適切なサイジング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こと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 (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 でオーバー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us-east-1。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、検出結果=過剰プロビジョニングでフィルタリングして、Fargate 上のすべての Amazon ECS サービスを表示します。
3. Fargate でオーバー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、「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レコメンデーションを保存するには、Compute Optimizer が各リージョンで に書き込むことができる 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「 の Amazon S3 バケットポリシー AWS Compute Optimizer」を参照してください。

Compute Optimizer からのレコメンデーションを表示するには、次の手順を実行します。

1. Compute Optimizer コンソール で、推奨事項のエクスポートページに移動します。
2. S3 バケットの送信先 で、S3 バケットを選択します。
3. エクスポートフィルターセクションのリソースタイプ で、Fargate の ECS サービスを選択します。
4. Fargate 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 からの CPU とメモリの推奨事項を確認します。例えば、「推奨タスクサイズと現在の設定の比較」セクションの推奨事項と、「推奨コンテナサイズと現在の設定の比較」セクションを参照してください。

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、次の手順を実行します。

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、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、適切なサイジングをターゲットにする予定の 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に保存します。

タグタスクの適切なサイジング

ワークロードにタグを付けることは、 でリソースを整理するための強力なツールです AWS。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、 を使用してAWS CloudShell、ターゲットアカウントおよび 内の任意の 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 AWS リージョン。

```
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="enabled"
# Get a list of ECS Clusters
ClustersArns=$( w secs list-clusters -query 'clusterArns' -output text)
for ClustersArn in $ClustersArns; do
 ServiceArns=$( w secs list-services -cluster $ClustersArn -query 'serviceArns' -output
 text)
 for ServiceArn in $ServiceArns; do
  TasksArns=$( w secs list-tasks -cluster $ClustersArn -service-name $ServiceArn -query
 'taskArns' -output text)
  for TasksArn in $TasksArns; do
    w secs 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達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
```

コスト配分タグによる AWS 請求ツールの使用を有効にする

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、ライツサイジングタグが認識され、 AWS 請求ツール (や など) AWS Cost Explorer で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については、 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。 AWS Billing and Cost Management

24 時間待ってから、次のセクションで適切なサイジングレコメンデーションを実装する前に、Cost Explorer で タグを確認できます。これを行うには、Cost Explorer で Rightsizing タグを検索します。

適切なサイジングのレコメンデーションを実装する

Compute Optimizer は、タスクまたはコンテナサイズのレコメンデーションを提供します。適切なサイジングレコメンデーションを実装するには、次の手順を実行します。

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。コンテナサイズのレコメンデーションを更新するには、ECS タスク定義 `memory` の containerDefinitions ブロックで `cpu`と を更新します。 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#task_size例:

```
"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、次の手順を実行します。

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:

```
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
```

コストの前後に確認する

リソースのサイズを適切に設定したら、Cost Explorer を使用して、適切なサイズ設定タグを使用してコストの前後に表示できます。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、ライツサイジングタグを使用して、ターゲットとなるすべての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の運用に伴う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 Rightsizing 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 1 日あたり 7.56 USD になります。1 か月あたり 744 時間と仮定すると、適切なサイジング前の年間コストは 11,044.9 USD です。適切なサイジングが完了すると、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。これを大規模な組織全体に与えた影響を想像してみてください。

適切なサイジング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、 に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 では、適切なサイズのレコメンデーションとライセンスガイダンスも提供されます。
- Savings Plans を購入する前に、すべての適切なサイズ設定を完了してください。これにより、Savings Plans コミットメントでの購入超過を回避できます。

次のステップ

次のステップを実行することをお勧めします。

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てください。
2. Savings Plansを確認します。

追加リソース

- 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)
- 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。唯一の固定コストと既知のコストは、Amazon Elastic Kubernetes Service (Amazon EKS) コントロールプレーンです。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可変量です。

Kubecost を使用して、名前空間とサービスから個々のポッド まで、Kubernetes インフラストラクチャのコストを分析し、ダッシュボードにデータを表示できます。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のコストと、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 out-of-cluster コストを表面化します。Kubecost は、このデータに基づいて適切なサイズのレコメンデーションを行い、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 と統合AWS

Cost and Usage Reportして、Compute Savings Plansリザーブドインスタンス、および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。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 ノードグループの適切なサイズ設定は、標準 EC2 インスタンスと同じ削減額を提供します。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、自動スケーリンググループに必要な 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削除できます。

コスト最適化に関する推奨事項

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. Kubecost を環境にデプロイする
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. 適切なサイズのクラスターノード
4. 適切なサイズのコンテナリクエスト
5. 使用率の低いノードを管理する
6. 「放棄されたワークロード」
7. レコメンデーションに基づく行動
8. セルフマネージド型ノードの更新

Kubecost を環境にデプロイする

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 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 AWS ブログ記事を参照してください。ま

たは、Kubecost のインストールと設定の手順については、公式の Kubecost ドキュメントを参照してください。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 ノードのサポート」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ことからもメリットを得られます。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、Savings を選択します。

適切なサイズのクラスターノード

Kubecost で、ナビゲーションバーから Savings を選択し、クラスターノード の適切なサイズを選択します。

Kubecost が vCPU と RAM の両方でクラスターが過剰にプロビジョニングされていることを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 VCPUs	10 VCPUs	8 VCPUs
RAM	152 GB	20 GB	18 GB
インスタンスの内訳	2 c5.xlarge + 2 その他	5 t3a.medium	c5n.large 2 個 + その他 1 個

Kubecost ブログ記事「Kubernetes クラスターに最適なノードセットを見つける」で説明されているように、シンプルなオプションは単一のノードグループを使用し、複雑なオプションはマルチノードグループアプローチを使用します。「導入方法を学ぶ」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。Kubecost クラスターコントローラー をインストールする必要があります。

eksctl によって作成されていないセルフマネージド型 Windows ノードを使用している場合は、「既存のセルフマネージド型ノードグループの更新」を参照してください。これらの手順は、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ています。

適切なサイズのコンテナリクエスト

Kubecost で、ナビゲーションバーから Savings を選択し、「適切なサイズのレコメンデーションをリクエストする」ページに移動します。このページには、ポッドの効率、適切なサイズのレコメンデーション、推定コスト削減が表示されます。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。

例として、Kubecost が CPU と RAM (メモリ) に関して一部のポッドがオーバープロビジョニングされていると計算したとします。次に、Kubecost では、新しい CPU 値と RAM 値に調整して、月間削減額の見積もりを達成することを推奨しています。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 Savings を選択し、使用率の低いノードの管理 を選択します。

このページで、クラスター内の 1 つのノードが CPU と RAM (メモリ) の点で十分に活用されていないため、ドレインして終了またはサイズ変更でき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に関する詳細情報が得られます。

「放棄されたワークロード」

Kubecost で、ナビゲーションバーから Savings を選択し、Abandoned Workloads ページを選択します。この例では、Windows という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に達しておらず、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た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、リソースを消費しない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、コストを節約できます。

レコメンデーションに基づく行動

「クラスターノードの適切なサイズ」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レコメンデーションを行います。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 の 2 種類があります。

セルフマネージド型ノードの更新

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。で作成されたノードグループは更新eksctlできないため、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ます。

例として、 (m5.2xlarge EC2 インスタンスng-windows-m5-2xlarge を使用する) という名前の Windows ノードグループがあり、ポッドを ng-windows-t3-large (コストを節約するために t3.large EC2 インスタンスによってバックアップされる) という名前の新しいノードグループに移行するとします。

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eksctl、次の手順を実行します。

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 () とも一致しますng-windows-m5-2xlarge。
3. ノードグループ を使用するようにデプロイを変更するにはng-windows-t3-large、ノードグループ を削除ng-windows-m5-2xlargeして を実行しますkubectl describe svc,deploy,pod -n windows。ノードグループが削除された時点で、デプロイはすぐ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状態であることを示しています。

5. ポッドがノードグループ で実行されていることを表示するにはng-windows-t3-large、`kubectl describe pod <pod_name> -n <namespace>`および `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「EKS セルフマネージド型ノードグループから EKS マネージド型ノードグループへのワークロードのシームレスな移行」では、オーバーサイズのインスタンスタイプの 1 つのノードグループから、ダウンタイムなしで適切なサイズの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を提供しています。

次のステップ

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 Savings ダッシュボードでレコメンデーション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 デプロイを確認することをお勧めします。AWS また、「コンテナブログ」のブログ記事「Kubecost collaborate to deliver cost monitoring for EKS customers」を参照してください。 AWS

追加リソース

- Amazon EKS ワークショップ (Amazon EKS ワークショップ)
- 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 (AWS ブログ)
- Amazon EKS Finhack Workshop (AWS Workshop Studio)
- 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および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、アプリケーションの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、Infrastructure as Code (IaC) テンプレートである必要な Amazon Elastic Container Service (Amazon ECS) と Amazon Elastic Kubernetes Service (Amazon EKS) のデプロイアーティファクトを生成します。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「App2Container の仕組みApp2Container 」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 を使用すると、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に留意すべき点がいくつかあります。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「Windows でサポートされているアプリケーション」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) バージョンと一致させます。このアプローチは、アプリケーションを基盤となる OS から切り離し、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 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でも、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、TomEEJBoss (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、 one-application-to-one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コストが最大 60% 削減されます。App2Container は、アプリケーションのコンテナ化プロセスを迅速化するために役立ちます。以下は、モダナイゼーションのニーズに合わせて App2Container を使用する利点の一部です。

- 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動することで、サポート終了に近づいているオペレーティングシステムに対処します。より新しいオペレーティングシステムに移行し、延長サポートの費用を回避し、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 レコメンデーション - コンテナへの移行 のコンテナの利点を確認します。
- アプリケーションの統合とコンテナ化は、コンピューティングリソースをより効率的に使用することで、コンピューティング、ストレージ、ライセンスのフットプリントを削減するのに役立ちます。
- 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高めることができます。

コスト最適化に関する推奨事項

App2Container の使用方法については、「 の開始方法 AWS App2Container」を参照してください。App2Container コマンドの詳細については、App2Container コマンドリファレンス」を参照してください。

次のステップ

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、ストレージのコストを削減し、アプリケーションオペレーターの運用オーバーヘッドを削減できます。

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 のイマージョンデーを設定するよう依頼してください。

追加リソース

- 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 がサポートするアプリケーション (AWS ドキュメント)
- Modernize with AWS App2Container Workshop (AWS Workshop Studio)
- AWS App2Container FAQs (AWS ウェブサイト)

[Storage (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事項です。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、計画に反映され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のストレージボリュームを保存できる、フルマネージド型のブロックストレージサービスです。Amazon のいくつかの機能を活用してEBS、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 を使用してワークロードに必要な正確な量IOPSとスループットをEBS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。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スナップショットの変更
- 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、本番稼働用および高性能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向けに汎用SSDボリュー

ムを提供しています。多くの AWS のサービス (Amazon を含むEC2) 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 SSDsと呼ばれる第 3 世代の汎用 は、2020 年 12 月にリリースされました。

gp3 の提供は、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 gp2 EBS 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結合されます。1 GB の容量ごとに、gp2 ボリュームは 3 IOPSつのパフォーマンスを取得します。つまり、2,000 GB の gp2 ボリュームは 6,000 個まで可能ですIOPS。gp3 ボリュームの場合、パフォーマンスはストレージ容量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 16,000 IOPSおよび 1,000 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 つの大きな変化は、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 から始まりますIOPS。これに対して、gp2 ボリュームは、同じパフォーマンス機能に到達する前に 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 へのアップグレードはパフォーマンスを大幅に向上させます。

最後に、gp3 ボリュームの料金は、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 ボリュームのコストの 20% で強化されたパフォーマンス機能をすべて提供します。

コストへの影響

パフォーマンスを容量から独立してスケーリングできるため、 IOPSとスループットを追加する際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、GiB - 月あたり 0.10 USD のプロビジョニングされた容量に基づいています。gp3 ボリュームの場合、料金は高性能プロビジョニングボリューム に似ています。このIOPSSSDボリュームには容量に 1 つのコストがかかり、追加IOPSとスループットに別のコストがかかります。

次の表で示すように、gp3 ボリュームの容量料金は GiB - 月あたり 0.08 USD (gp2 より 20% 安価) で、個別のコストは IOPSのプロビジョニング済み IOPS- 月あたり 0.005 USD が 3,000 を上回り、プロビジョニング済み MiBs- 月あたり 0.04 USD がスループット MiBs 用です。

	gp3	gp2
ボリュームサイズ	1 GiB – 16 TiB	1 GiB – 16 TiB
ベースライン IOPS	3,000	3 IOPS/GiB (最小 100 IOPS) ~ 最大 16,000 IOPS

	gp3	gp2
		1 TiB 未満のボリュームは最大 3,000 までバーストできます IOPS
最大 IOPS/ ボリューム	16,000	16,000
ベースラインスループット	125 MiBs	スループット制限は MiBs、ボリュームサイズに応じて 128 MiBs ～250 です。
最大スループット/ボリューム	1,000 MiBs	250 MiBs
価格	0.08 USD/GiB - 月 3,000 件以上のIOPS無料 3,000 件、0.005 USD/プロビジョニング済み IOPS- 月 125 を上回って 1 MiBsか月あたり 125 MiBs 無料、0.04 USD/プロビジョニング MiBs	月あたり 0.10 USD/GiB

> **Important**
>
> gp3 ボリュームには容量とパフォーマンスのコストが別途ありますが、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、gp2 ボリュームよりも常に安価です。

次の表は、さまざまな容量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 (MiBs)	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 (MiBs)	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 (MiBs)	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 (MiBs)	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リソース EBS の「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。計算ツールをダウンロードし、それを使用して gp2 ボリュームを gp3 に移行することで節約できる量を確認できます。

コスト最適化に関する推奨事項

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。

追加リソース

- 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 クラウド運用と移行ブログ)
- 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)
- 2020 年の re:Invent 発表からのコスト削減の発見 (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 への移行コスト削減計算ツール (ダウンロード)

Amazon EBSスナップショットの変更

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。 point-in-time スナップショットを作成することで、EBSボリュームのデータを 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あるため、最新のスナップショットの後に変更されたデバイスのブロックのみを保存します。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、データを新しいEBSボリュームに (スナップショットが作成された時点から) 復元するために必要なすべての情報が含まれています。

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スナップショットの保持期間に対して課金されます。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、保存されている変更されたブロックに対してのみ課金されます。アーカイブ階層の場合、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課金されます。また、アーカイブ階層 からスナップショットを取得するためにも課金されます。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 のデータを保存してい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課金されます。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。その後、増分スナップ B の追加ストレージ 5 GB に対してのみ課金されます。
- 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 属性の値が、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`False`では に設定されます。これにより、オペレータが

EC2インスタンスを削除しようとしたが、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を残す状況が発生します。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を表示する」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、そのスナップショットは Amazon EBS Snapshot Standard 階層 (標準階層) に保存されます。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 EBSSnapshots Archive は、アクセス頻度の低いスナップショットや高速取得を必要としな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の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のスナップショットストレージコストが最大 75% 削減されます。

Amazon EBS スナップショットストレージ	コスト
標準	0.05 USD/GB/月
アーカイブ	0.0125 USD/GB/月

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TBsのEBSスナップショットがあるEC2インスタンスが存在する大規模な では、節約額がより大きくなります。

次の表は、1 か月あたりの標準階層とアーカイブ階層の使用量をわずか 50 TB で比較したものです。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
スタンダード 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

コスト最適化に関する推奨事項

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 のデータを変更して 2 番目のスナップショットを作成する場合、2 番目のスナップショットのサイズは 4 GiB です。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、「 AWS ナレッジセンター」の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。

以下の点を考慮します。

- 別の が AWS アカウント 所有し、自分の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請求されません。共有スナップショットを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課金されます。
- 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) から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のみが削除されます。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課金されます。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「増分スナップショットの削除」を参照してください。

でワークロードを実行する場合、スナップショットの冪等性は標準的な運用方法です AWS。時間の経過とともに、スナップショットは不要なデータに対して合計してコストがかかる可能性があります。

追加リソース

- 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)
- Amazon EBSスナップショットの削除 (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)

- Amazon Data Lifecycle Manager EBS を使用して Amazon スナップショットを自動的にアーカイブする (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

ボリュームがアタッチされていない (孤立している) と、 AWS 環境に不要なストレージコストが発生するEBS可能性があります。使用していないボリュームや使用されていないEBSボリュームの定期的なレビューと削除は、 AWS 環境の健全性の一環として組み込むことが不可欠です。EBS ボリュームの使用状況を継続的に確認するためのプロセスを設定するのがベストプラクティスです。を使用してAWS Compute Optimizer、使用率の低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ボリュームや十分に活用されていないEBSボリュームを特定、管理、削除します。

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの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 は永続的ストレージを提供し、EC2インスタンスとのアタッチとデタッチを柔軟に行うことができます。つまり、EC2インスタンスが終了しても、EBSボリュームのライフサイクルは保持されます。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`True`の 属性は に設定され、削除されます。他のボリューム`False`では に設定され、保存されます。

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した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 のEBS料金の平均コストは 1 GB あたり 0.10 USD と最小限に見える場合がありますが、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多大なコストが発生す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 50 個の未使用のEBSボリュームを保持することによる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 USDUSD)	100 GB	100 GB 50.00 EBS ボリューム月 $0.10 USD = $500.00 USD

前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 のコスト削減が得られます。これはコスト削減のための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、通常のプラクティスとして AWS 環境の健全性に組み込んでください。

コスト最適化に関する推奨事項

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 、および を使用して CloudWatch、経過時間、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、添付されていない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ボリュームの管理

以下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。
- データのバックアップポリシーと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にバックアップします。データ保持の場合、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方法でデータを保持し、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スナップショットの変更」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 リソースの間に依存関係がないか確認します。AWS Management Console 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、追跡を容易にするためのタグ付けメカニズムを適用し、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ボリュームのタグ付け — 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プラクティスです。 EBSこれは、どの未アタッチボリューム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。
- 安全な削除の確認 — EBSボリュームが最後にアタッチされた日時を確認すると、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 ナレッジセンターの 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方法」を参照してください。
- 使用率の低いEBSボリュームの特定と削除 — 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 また、 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、効率を向上させるためのレコメンデーションを提供するのに役立ちます。例えば、「 (GitHub) による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)」、「 を使用した Amazon EBSボリュームのコスト最適化 AWS Compute Optimizer」 (AWS ストレージブログ) 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) – AWSSDKs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検索できます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。のサンプルスクリプト GitHub は、この仕組みを示しています。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、オートメーションランブック を作成します。このランブックは、特定の順序で実行する一連のステップを定義します。例えば、まず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、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 リソースの変更を経時的に評価、監査、追跡できます。設定の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。例えば、 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 AWS Config て、未使用のEBSボリュームの削除を自動的に修正できます。

追加リソース

- AWS Config および 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 オートメーション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型の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、カスタム 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 SharePointしてアクセスできます。NETさらに、 FSx for Windows File Server は、 pay-as-you-go 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、コストの管理に役立ちます。

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 でデータ重複排除を有効にする FSx
- FSx for Windows File Server のデータシャーディングを理解する
- Amazon でのHDDボリューム使用量を理解する FSx
- 1 つ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のイノベーションとセキュリティを組み合わせながら、業界トップの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。サービスをコードとしてのインフラストラクチャ (IaC) ワークフローに組み込み AWS 、 AWS コンピューティング、モニタリング、データ保護サービスと統合できます。Windows ワークロードでは、アプリケーションのニーズに応じて使用できる 2 つのフルマネージドファイルサービスから選択できます。Windows File Server FSx の場合は 、Amazon FSx for の場合は です NetApp ONTAP。

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型の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、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FSxには Windows File Server の を使用し、SQLサーバー、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 () 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VDI。FSx for Windows File Server は、Amazon FSx File Gateway、Amazon Kendra、Amazon S3 の監査ログ、および Amazon Data Firehose とも適切に統合されています。

ONTAP に関する FSx

FSx の 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、主に既存のオンプレミス NetApp ユーザーに推奨されます。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サーバー用 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、64 TB を超えるファイルシステム (DFS名前空間サーバーのない 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) 、およびデータのインテリジェントな階層化など、複数のプロトコルをサポートします。

コストへの影響

FSx for Windows File Server

FSx for Windows File Server は、SQLサーバー用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スタンダードエディション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サーバーエンタープライズエディションの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サーバーライセンスを 65～75% 節約できます。

for FSx Windows File Server for Failover Cluster Instances を使用して、ストレージ I/O を一般的なEBSストレージからオフロードできます。Windows File Server FSxの に I/O をオフロードすることで、ストレージスループットに影響を与えIOPSずに、Amazon の高EBSスループットと に依存するEC2インスタンスをスケールダウンできます。

ONTAP に関する FSx

FSx の 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し、SQLサーバーインスタントファイル初期化、 を使用したクロスリージョンレプリケーション SnapMirror、ウイルス対策サポート、クローン作成のメリットを受けることができます。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、 を使用して NetApp SnapCenter、 の を使用して Server のEC2インスタンスのバックアップ、復元SQL、クローン機能を管理するFSxことができますONTAP。FSx の は、パフォーマンスとコスト効率を混在させるために、 から低コストの容量プールストレージSSDへの自動階層化ONTAPも提供します。

FSx for ONTAPは、Windows ネイティブ NetAppファイルシステムをサポートする FSx Windows File Server とは異なり、 のNTFSファイルシステム (ONTAP) をサポートします。FSx の最小サイズ ONTAPは 1024 GB ですが、Windows File Server FSxの は最低 32 GB で起動できます。

Microsoft 分散ファイルシステムとの統合

FSx for Windows File Server および FSx for は、Microsoft の分散ファイルシステム (DFS) と ONTAP統合して、既存のデプロイにシームレスに統合できます。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 Windows File Server 用の と FSx用の 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DFSDFSN) で名前空間 () ONTAPをサポートします。

- Windows File Server FSxの のみがレDFSプリケーション (DFSR) をサポートし、単一のアベイラビリティーゾーンを使用する場合にのみサポートされます。

コスト最適化に関する推奨事項

Windows File Server と FSxの両方のパフォーマンスFSxONTAPは、料金と同様に、設定によって大きく異なります。FSx for Windows File Server の料金は、主にストレージ容量とストレージタイプ、スループット容量、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。for ではFSxONTAP、SSDストレージ、、キャパシティープールの使用量SSDIOPS、スループットキャパシティー、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	構成	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)
ONTAP に関する FSx	979.28 USD	単一のアベイラビリティーゾーン 100% SSD 15,000 読み込み/書き込みキャパシティー階層 15,000 SSD IOPS 128 MBps	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	構成	リージョン
		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 の を使用してデータをONTAP階層化できます。100% のデータに定期的にアクセスし、SSD ストレージを必要とすることは非常にまれです。コールドデータとアクセス頻度の低いデータをキャパシティー層に移動して、コスト削減を実現できます。
- ここで説明する料金は、 階層の 100% のデータと SSD 階層の 15,000 SSD IOPSのデータで計算されます。

バックアップ

デフォルトでは、Windows File Server FSxの ONTAPと の両方FSx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 では、 を使用してバックアップするための追加オプションONTAPがあります。これにより SnapVault、容量層内に存在するようにバックアップを設定できます。によるバックアップ SnapVault 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、1 GB あたり 0.05 USD です。FSx の SnapVault バックアップ ONTAP (容量プールストレージ SSDへの 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 時間の精度を提供します。SnapVault では、最低 5 分まで短縮できます。
- NetAppのツール (CLIや などAPI) を使用して、 SnapVault 関係とスナップショットの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、容量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 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、1 つ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オーバープロビジョニングの防止により、コストを節約し、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、次の操作を行います。

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、Performance Monitor 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. Performance Monitor では、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 の概要」を参照してください。

4. 正しい設定を特定したら、Microsoft 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りが正しいかどうかをテストしますDISKSPD。
5. パフォーマンスに満足している場合は、ファイル共有にカットオーバーします。

ストレージ容量はスケールアップのみできるため、控えめにアプローチすることをお勧めします。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およびスケールダウンできます。

追加リソース

- 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 でデータ重複排除を有効にする FSx

概要

データ重複排除は、データをより効率的に、より少ない容量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、データ内の重複を見つけて削除することが含まれます。データ重複排除では、サブファイル可変サイズのチャンク化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 の最適化率が得られます。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は、ハードウェア障害時の回復性があります。メタデータの冗長性や最もアクセス頻度の高いデータチャンクなど、データとメタデータに対する完全なチェックサム検証があります。

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、削減額はユーザードキュメントで 30～50%、ソフトウェア開発データセットで最大 70～80% の範囲です。データ重複排除で実現できるストレージの節約は、ファイル間での重複量など、データセットの性質によって異なることを理解することが重要です。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

エンタープライズ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、容量スケーリングとデータ最適化の主要な目標を作成します。データ重複排除のデフォルト設定により、すぐに節約できます。また、管理者が設定を微調整して、追加のメリットを確認することもできます。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。ジョブは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利用できます (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率は 25% にすぎませんが、 FSx for Windows File Server では大幅なコスト削減が可能です。これらの削減額は、「」の見積りに基づいています AWS Pricing Calculator。

コスト最適化に関する推奨事項

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。でリモート管理 PowerShell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、 for Windows File Server ドキュメントの「ファイルシステムの管理FSx」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
}
```

コマンドレットを実行する PowerShell `Measure-DedupFileMetadata`ことで、フォルダのグループ、単一のフォルダ、または単一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、ボリュームで再利用可能なディスク容量を決定できます。具体的には、これらのファイルを削除するとどれだけの容量が戻るかが `DedupDistinctSize`の値によって決まります。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ことが多いため、重複排除エンジンはどのチャンクが一意で、ガベージコレクションジョブの後に削除されるかを計算します。

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機能し、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度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を実行するように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 を使用して増やすことができます-memory switch。最適化ジョブでは、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 (通常はオフ時間で実行するようにスケジュール) では、メモリ消費量を増やすことができます (50 など) 。

データ重複排除設定の詳細については、 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FSx」を参照してください。

追加リソース

- 「データ重複排除について」(Microsoft ドキュメント)
- データ重複排除によるストレージコストの削減 (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、ストレージ容量、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容量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、主にストレージ容量とストレージタイプ、スループット容量、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では、ファイルデータを小さなデータセット (シャード) に分割し、異なるファイルシステム間で保存します。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。さらに、ファイルデータストレージを、各ファイルシステムがサポートするサイズ (64 TB) を超えて、数百ペタバイトまで拡張することもできます。

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、Windows File Server ファイルシステムFSx用に複数の小さな をデプロイする方

が効果的です。FSx for Windows File Server HDDとSSDストレージタイプを組み合わせて使用すると、コスト削減が向上し、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と一致させることができます。次の表では、1 つの 17 TB ファイルシステムの違いを確認し、同じ容量に を追加する複数の小さなファイルシステムと比較します。

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)	コスト	構成	リージョン
Amazon FSx for Windows File Server	5,716 USD USD	17 TB SSD 重複排除 30% 256 Mbps 17 TB バックアップ	米国東部 (バージニア北部)

を使用したパーティション化されたワークロード DFSN

[Server name] (サーバー名)	コスト	構成	リージョン	共有
Amazon FSx for Windows File Server	1,024 USD USD	2 TB SSD 重複排除 20% 128 Mbps 2 TB バックアップ マルチ AZ	米国東部 (バージニア北部)	共有 1
Amazon FSx for Windows File Server	2,132 USD USD	5 TB SSD 重複排除 30% 256 Mbps	米国東部 (バージニア北部)	共有 2

[Server name] (サーバー名)	コスト	構成	リージョン	共有
		5 TB バックアップ マルチ AZ		
Amazon FSx for Windows File Server	1,036 USD 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 USD	t3a.medium 2 vCPUs 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と照合しながら、26% の節約を実現できます。料金見積りの詳細については、見積りを参照してくださいAWS Pricing Calculator。

コスト最適化に関する推奨事項

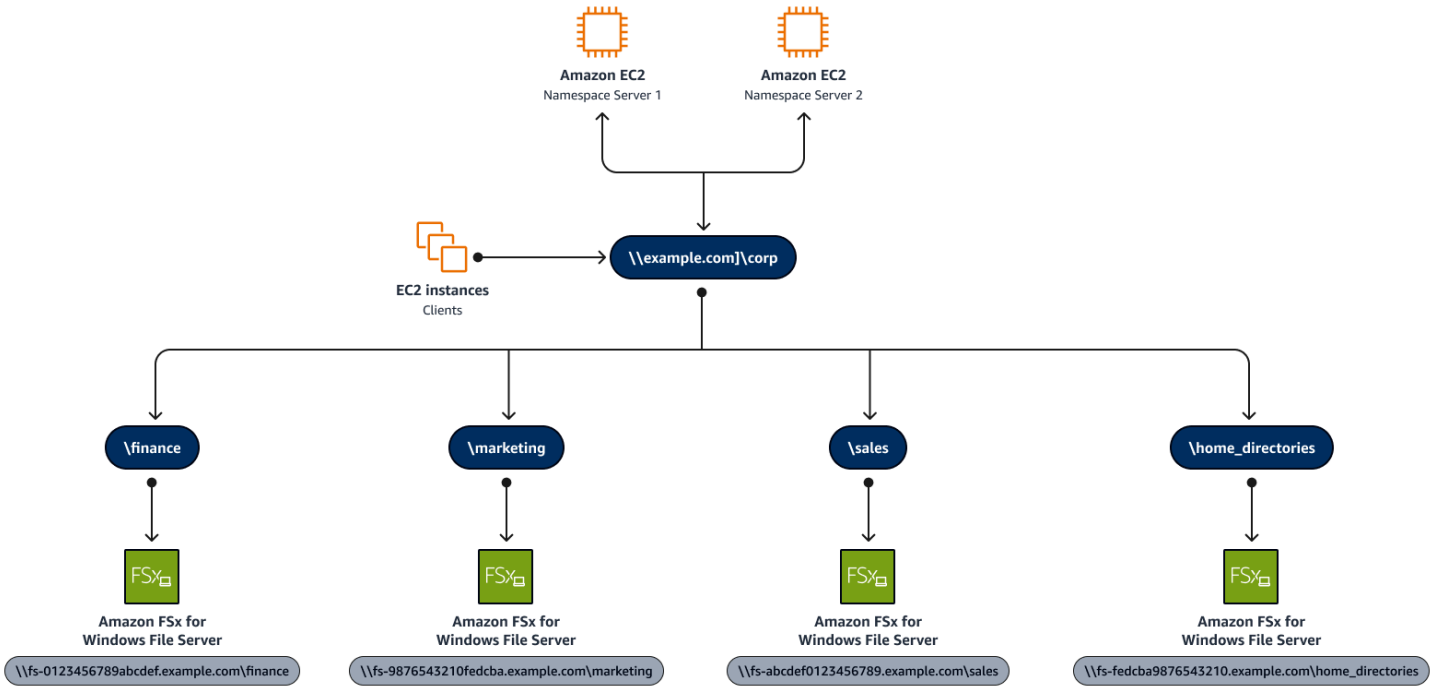
データ重複排除ソリューションをデプロイするには、データタイプ、I/O サイズ、I/O アクセスパターンに基づいて Microsoft DFS名前空間を設定する必要があります。各名前空間は、最大 50,000 のファイル共有と数百ペタバイトのストレージ容量を集約してサポートします。

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規則を選択するのが最も効率的です。ワークロードをモニタリングすると、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関するヘルプが必要な場合は、

「 for FSx Windows File Server ドキュメント」の「 for Windows File Server performanceFSx」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でさまざまなファイルシステムにアクセスしているときに、1 つの同種ファイルシステムを表示できます。エンドユーザーが共有の設計対象ワークロード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、本番共有と非本番共有にラベルを付けることも重要です。

次の図は、1 つの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。

- Windows File Server 共有FSx用の既存の DFS をツリーに追加できます。
- Amazon FSx を共有DFS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 名前空間設定を処理するには、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「名前空間の概要」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、 の「Amazon FSx for Windows File Server でDFSの名前空間の使用」ビデオを参照してください YouTube。

追加リソース

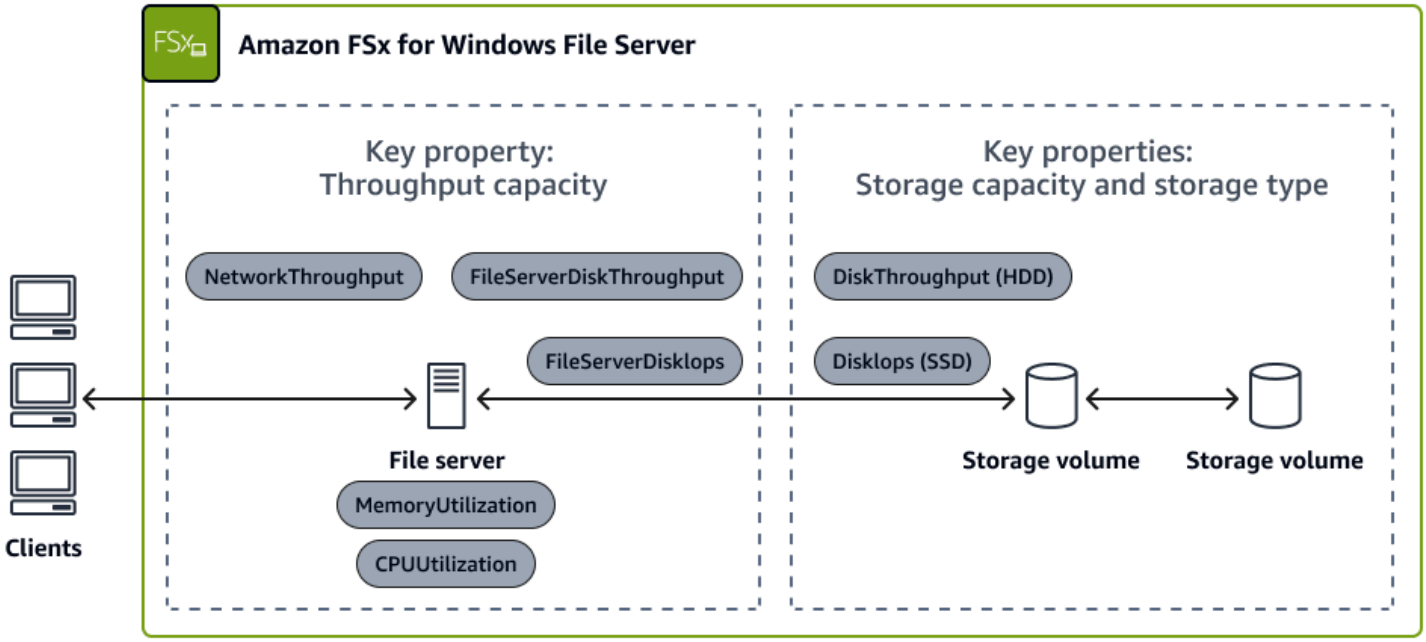
- 複数のファイルシステムをDFS名前空間でグループ化する (Amazon FSxドキュメント)
- チュートリアル 6: シャードによるパフォーマンスのスケールアウト (Amazon FSxドキュメント)
- Amazon FSx for Windows File Server でDFSの名前空間の使用 (AWS ラボ)

Amazon でのHDDボリューム使用量を理解する FSx

概要

Amazon FSx for Windows File Server では、ファイルシステムの容量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) と HDD (ソリッドステートドライブ) の 2 SSD つの容量設定を使用できます。EBS st1 ドライブは、 のファイルシステムストレージに使用されますHDD。EBS io1 ドライブは に使用されますSSD。

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 (IOPsストレージの TiB あたり) の 12 IOPSベースラインと、80 バーストメガバイト/秒 (ストレージの TiB あたり) の 12 MB/秒のスルー

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 の両方のベースラインとして 50 * 12 = 600 が得られますIOPS。

Amazon FSx for Windows File Server には 80 バースト が用意されていますIOPS。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超えると自動的に消費されます。例えば、ワークロードが 10 IOPS/TB を 1 時間しか利用していない場合 (ベースラインレートより 2 IOPS/TB 低い場合) 、バーストクレジットが再び不足する前に、次の 1 時間 14 IOPS/TB (ベースラインより 2 IOPS/TB 高い場合) を利用できます。

ファイルオペレーションの場合、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 桁ミリ秒のレイテンシーを提供します。HDD ストレージを含むすべてのファイルシステムについて、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る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と、ストレージ容量全体のコストを削減し、ニーズに合った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、ストレージ容量、ストレージタイプ、スループットの 3 つの要素によって異なります。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 集約型ワークロードには 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せるワークロードはさまざまです。HDD ストレージは、ホームディレクトリ、ユーザーと部門の共有、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、保存しているほとんどのデータへのアクセスはまれです。

を使用してAWS Pricing Calculator、20 TB SSDと のHDDファイルシステムを比較できますus-east-1。次の表に示されているように、重複排除による節約がなくても、HDDファイルシステムとSSDファイルシステムを比較する場合、コスト差は大きくなります。

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**
>
> Windows File Server FSxのその他の節約については、このガイドの「Amazon でデータ重複排除を有効にするFSx」セクションを参照してください。

パフォーマンスの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、コストを削減できます。

コスト最適化に関する推奨事項

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していることを確認します。HDD ストレージはストレージに比べて低コストですが、SSDストレージ単位IOPSあたりのディスクスループットとディスクレベルは低くなります。I/O 要件の低い汎用ユーザー共有やホームディレクトリ、データの取得頻度の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が少ないデータセットに適しています。

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、 の最小容量HDDがはるかに高い 2 TB であることに注意してください。

別のストレージタイプのバックアップを復元するには、次の手順を実行します。

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. 必要な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が正しいストレージタイプであり、データがそのままであることを確認してください。

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、変更が許容できること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「新しいメトリクスによる 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。

追加リソース

- 「Amazon によるコストの最適化FSx」(Amazon FSxドキュメント)

1 つ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、Amazon FSx for Windows File Server をマネージドファイルストレージサービスとして使用できるようにするシナリオについて説明します。本番ワークロードFSxには、Amazon 用に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

1 つ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では、1 つのアベイラビリティーゾーンファイルシステムで GB 月あたり 0.230 USDSSD、 GB 月あたり 0.025 USD HDDを支払うのに対しSSD、 HDDでは GB 月あたり 0.130 USD、 では GB 月あたり 0.013 USD を支払うことになります。を使用して、コストの比較を確認し、独自の見積りを作成できますAWS Pricing Calculator。

10 TB ファイルシステムの場合、複数のアベイラビリティーゾーンに対して 1 か月あたり約 1,200 USD、または 1 つのアベイラビリティーゾーンに対して 1 か月あたり 680 USD を支払うことの違いが考えられます。この例では、 で 10 TB FSx の Windows File Server ファイルシステムを使用していますSSD。重複排除の推定削減額は 50% です。全体として、1 つ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。

コスト最適化に関する推奨事項

単一アベイラビリティーゾーンのデプロイ

1 つ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、 for Windows File Server FSx に保存されているデータSLAsについて、独自の内部 を考慮してください。これには、顧客 (内部および外部) に SLAs を提供する必要があるかどうか、および Amazon FSx1 つのアベイラビリティーゾーンの 9 つのアベイラビリティーでそれらの を満たすことができるかどうかについての理解が必要ですSLAs。FSx 1 つのアベイラビリティーゾーンを持つ Windows File Server の稼働時間は、99.9% です。複数のアベイラビリティーゾーンFSxの SLA Amazon の 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では、追加料金がかかる場合でも、1 つのアベイラビリティーゾーンで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。

単一アベイラビリティーゾーンのデプロイは、SQLサーバー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、一貫した稼働時間を実現できます。可用性の高いSQLサーバーや本番稼働用アプリケーションへのアクセスなど、本番稼働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は、1 つのアベイラビリティーゾーンがワークロードに適していません。バックアップ、非本番環境のテスト、開発環境では、Amazon 1 FSxつのアベイラビリティーゾーンを実装することで運用コストを削減できます。

Amazon FSx1 つの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 1 つのユースケースは、Always On 可用性グループを使用する高可用性SQLサーバー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 FSx1 つの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サーバー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

単一のアベイラビリティーゾーンファイルシステム (単一の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システム) でコストを削減するための潜在的なオプションは、Amazon との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FSx。ネイティブ Microsoft -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。 DFSAmazon を使用して DFS-R を設定する方法の詳細についてはFSx、Amazon FSxドキュメントの「Microsoft 分散ファイルシステムレプリケーションの使用」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 ファイルゲートウェイを別のリージョンに実装して、Amazon のマルチリージョンアクセスが可能になりますFSx。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、リージョン間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。リージョン間でトラフィックを移動すると、0.02 USD/Gb の料金が発生します。したがって、大量のデータに一貫した変更がある場合、全体的なコストが増加します。例えば、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 で単一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が重要な考慮事項ですFSx。メンテナンス期間中、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、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。

追加リソース

- 可用性と耐久性: シングル AZ およびマルチ AZ ファイルシステム (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 を使用して、 や などのネイティブ Windows プロトコルを使用してデータを保存SMBおよびアクセスできますNFS。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に大きな変更を加える AWS 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 の傘下に、Amazon S3 ファイルゲートウェイ、Amazon FSx ファイルゲートウェイ、テープゲートウェイ、ボリュームゲートウェイがあります。S3 File Gateway と FSx File Gateway は、Microsoft ワークロードで最もよく使用されます。

Amazon S3 File Gateway

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Amazon S3 にファイルを保存できます。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 を使用して Storage Gateway を実装すると、ライフサイクルファイルを低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、高性能または高可用性の書き込み、ファイルの編集、または部門共有にはお勧めしません。

Amazon FSx File Gateway

Amazon FSx File Gateway では、Amazon FSx Windows ファイルシステムを使用する際のコスト削減も可能です。FSx ファイルゲートウェイを立ち上げて、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、2 つの独立したファイルシステムを持つコストを回避できます。これは、オンプレミスのファイルサーバーが複数あり、複数のハードウェアデバイスに料金がかからないように統合する場合にも役立ちます。

コストへの影響

Amazon S3 File Gateway

Storage Gateway の起動ウィザードを使用できるため、S3 File Storage Gatewayの設定は簡単です。 AWS 環境内のEC2インスタンスを使用して、ゲートウェイを数分でデプロイできます。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、NTFSアクセスコントロールリスト (ACLs) 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 File Gateway
- 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 のストレージをかなり安価に保存できます。料金見積りでは、S3 Intelligent Tiering が料金の柔軟性に使用されます。これには、S3 Standard では 80%、低頻度アクセスでは 10%、Amazon S3 Glacier では 10% が

含まれます。S3 Glacier を使用できますが、S3 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。S3 Glacier はアーカイブでの使用のみを目的としており、通常のアクセス使用ではありません。

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 USD SSD	米国東部 (バージニア北部)
Amazon S3 File Gateway	USDインテリジェント階層化 449.51 USD	米国東部 (バージニア北部)
Amazon EBS	1,335.69 USD USD GP3	米国東部 (バージニア北部)

以下の点を考慮します。

- S3 Glacier では、 を使用してオブジェクトRestoreObjectAPI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、オペレーションチームは、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「エラー： InaccessibleStorageClass」を参照してください。
- S3 Glacier のアクセス制限に加えて、Storage Gateway のオブジェクト/フォルダごとにACLs許可されるのは 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 File Gateway

Amazon S3 File Gateway と同様に、FSxファイルゲートウェイはデータを長期的に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 ファイルゲートウェイでは、データは Amazon S3 にあります。FSx File Gateway の場合、データは for Windows File Server FSx にあります。for Windows File Server FSxではマルチ AZ オプションを使用できますが、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、エンドユーザーに地理的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かかります。まったく新し

いファイルシステムの作成とコストの重複を避けるために、FSxファイルゲートウェイ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、ユーザーにファイルへのローカライズされたアクセスが提供され、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 USD SSD	米国東部 (バージニア北部)
Amazon FSx File Gateway	503.70 USD/シングルゲートウェイ	米国東部 (バージニア北部)

> **Note**
>
> 前の表の料金は、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の月額約 180 USD (または年間 2,100 USD) を節約できます。
- FSx ファイルゲートウェイでは、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。
- FSx for Windows File Server の 2 つのデプロイを異なる リージョンに作成し、 AWS Backup または で更新しておくことはできますが 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。

コスト最適化に関する推奨事項

Amazon S3 File Gateway

S3 File Gateway にはファイルを保存するための低コストのオプションがありますが、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EC2を使用して Amazon 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したい場合は、Storage Gateway を VMwareや Hyper-V などの仮想化プラットフォームに仮想アプライアンスとしてデプロイできます。

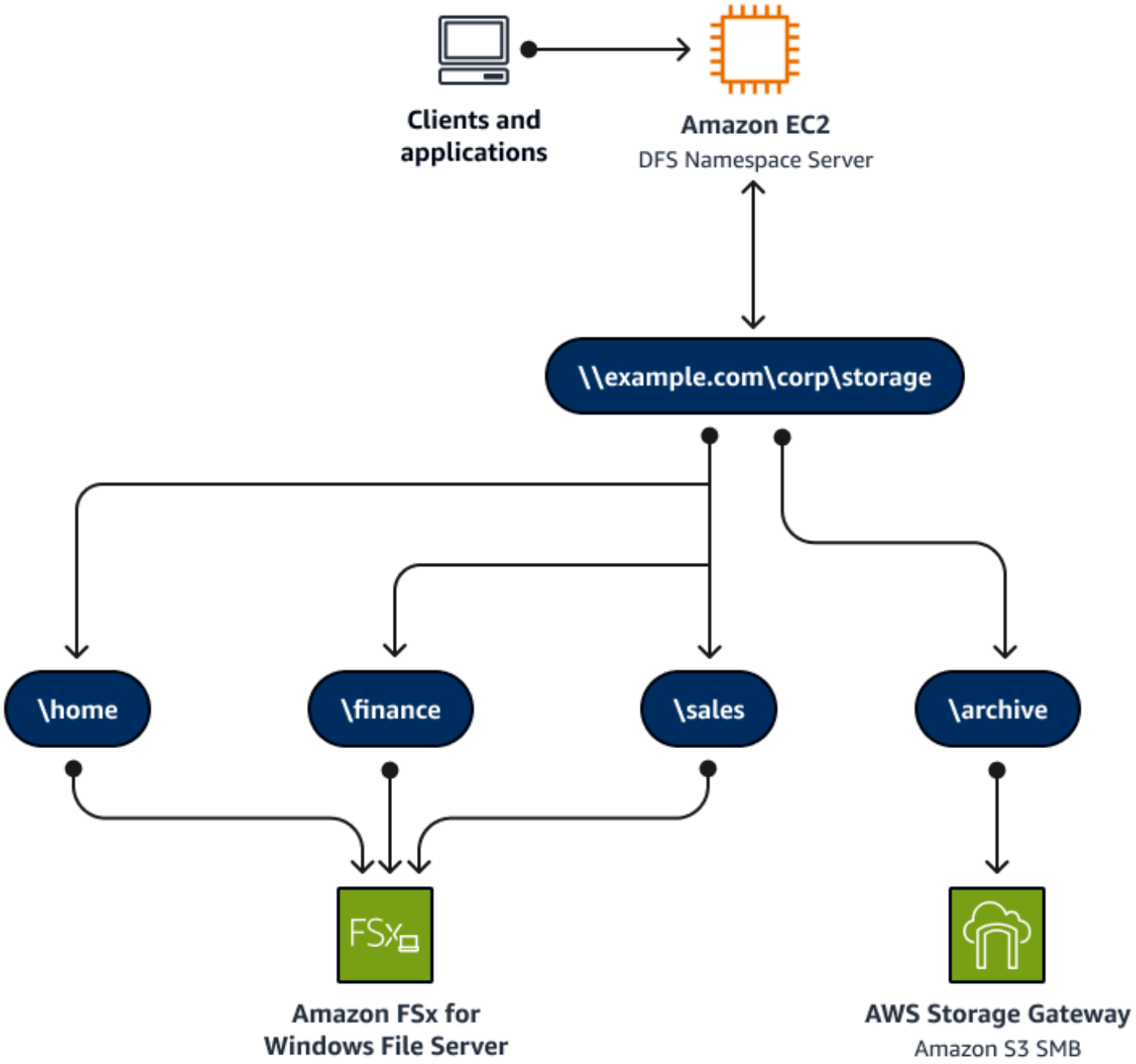
高可用性に関する考慮事項

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または停止および起動できる厳格なアクセスコントロール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 の実装を迅速に復旧することをお勧めします。を使用してオンプレミスで Storage Gateway を実行している場合は VMware、高可用性 に設定できます。

複数のファイルシステムの実行

日常的に使用する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ことで、不要なストレージコストを回避できます。Storage Gateway は、 for FSx 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、 for FSx Windows File Server で実行されているプライマリの日次使用ストレージとAmazon S3 で実行されているストレージ (Storage Gateway 経由でアクセス) を表示できます。

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誘導されます。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することも、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。

以下の点を考慮します。

- 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 を使用してファイル ACLs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「ステップ 3: Amazon File Gateway を作成してアクティブ化する」を参照してください。 FSx環境に FSx File Gateway をデプロイした後、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できます。

ストレージは、FSxファイルゲートウェイ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、ファイルをキャッシュするための十分な容量です。空き領域が少ない場合のモニタリングアラートを作成すると、ストレージの適切なサイジングに過剰に割り当てることなく役立ちます。

追加リソース

- AWS Storage Gateway リソース (AWS ドキュメント)

アクティブディレクトリ

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、使用量に対してのみ料金を支払うと同時に、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。Active Directory サービスは、Windows Server 環境でのアイデンティティ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 のセルフマネージド Active Directory
- AWS Managed Microsoft AD
- AD Connector

Amazon EC2 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提供します。主な目的は、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インスタンスファミリーインスタンスを選択する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」	コスト
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」	コスト
m5a.large	2	2	8	259.88 USD/月

コストの詳細については、「 の AWS Pricing Calculator 見積り」を参照してください。

1 か月あたり 178.12 USD の節約により、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。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に、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。

コスト最適化に関する推奨事項

Microsoft は、Active Directory 環境をデプロイする場合の容量計画に関する推奨事項を提供します。Active Directory 環境を計画またはスケーリングするときは、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」
- ネットワーク
- [Storage (ストレージ)]
- 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、 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 AWS。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 AWS、 でオンプレミス環境をレプリケートする必要がないことが明確になります。

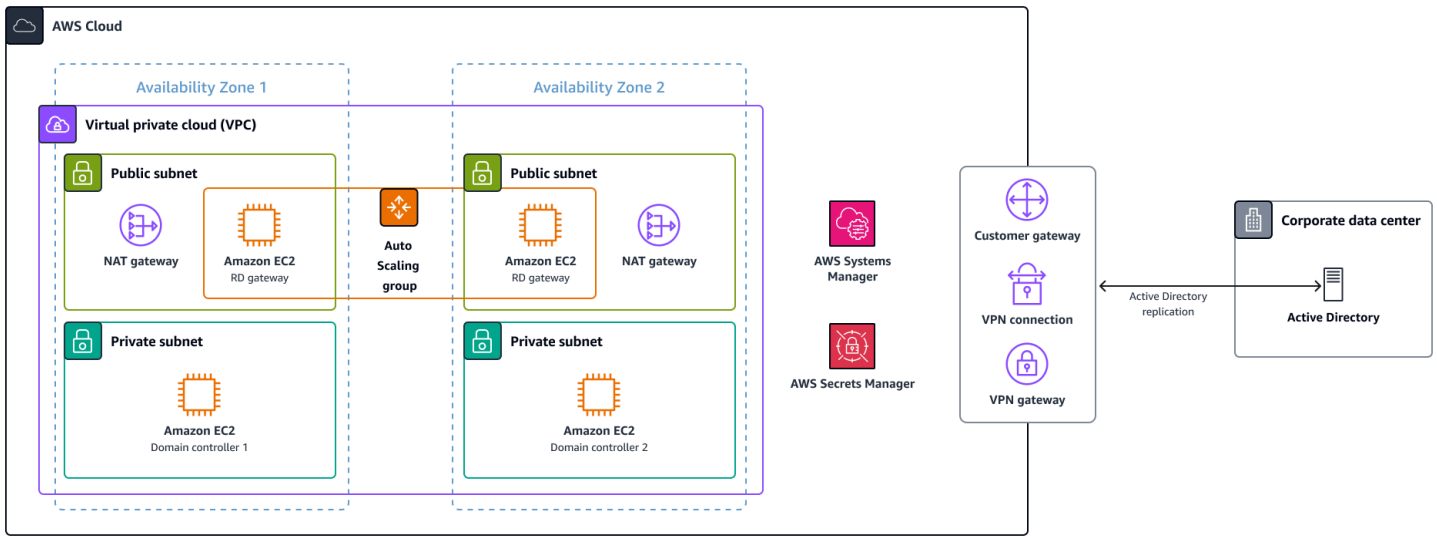
次の表は、 AWS フットプリントの vCPU、メモリ、ディスクに関する重要なコンポーネントを示しています。

コンポーネント	見積り
ストレージ/データベースサイズ	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
ネットワーク	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 に展開します AWS クラウド。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 にデプロイされるわけではありません AWS。そのため、移行作業のコストを節約するために、最初は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。

サーバーとユーザーがオンプレミスで認証するオンプレミスフットプリントを維持する場合、 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 AWS。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、オンプレミスの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しながら、 のドメインコントローラーに対してのみフットプリントを認証できます AWS AWS。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ではなく、 AWS リソースのみの使用に制限 AWS することで、 での Active Directory のフットプリントの過剰サイズを回避できます。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「ドメインコントローラーの適切な配置」と「サイトに関する考慮事項」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 用に 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、インスタンスタイプを簡単に変更して、 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 AWS。このガイドの「Windows on Amazon EC2」セクションでは、 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ます AWS。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 に完全に移行する場合は AWS、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。オンプレミスにあるものを に複製する前に 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。Performance Monitor 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してください。これにより、全体的な使用状況を把握できるため、適切なサイズで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、使用率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、支出を削減することも重要です。を使用して AWS Compute Optimizer 、 で実行している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 ワークロードを適切にサイズ設定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、Performance Monitor を使用して 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、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、 で利用可能なメトリクスを使用して CloudWatch 、適切な意思決定を行うことができます。エージェントを使用して CloudWatch、データ収集のために送信されるカスタム Performance Monitor メトリクスを設定できます。手順については、「 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。
AWS

CloudWatch エージェントをデプロイしたら、 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metrics_collected。

メトリクスカテゴリ	メトリクス名
Database to Instance (NTDSA)	データベースキャッシュ % ヒット
I/O データベースの読み取り平均レイテンシー	
I/O データベースの読み取り/秒	
I/O ログ書き込みの平均レイテンシー	
DirectoryServices (NTDS)	LDAP バインド時間
DRA 保留中のレプリケーションオペレーション	
DRA 保留中のレプリケーション同期	
DNS	再帰クエリ/秒
再帰クエリの失敗/秒	
TCP クエリ受信/秒	
受信したクエリの合計/秒	
送信されたレスポンスの合計/秒	
UDP クエリ受信/秒	
LogicalDisk	平均ディスクキュー長
% 空き領域	
メモリ	使用中のコミットバイト数の割合
長期平均スタンバイキャッシュ有効期間 (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	送信バイト数/秒
Bytes Received/sec	
現在の帯域幅	
NTDS	ATQ 推定キュー遅延
ATQ リクエストのレイテンシー	
DS ディレクトリの読み取り/秒	
DS ディレクトリ検索/秒	
DS ディレクトリの書き込み/秒	
LDAP クライアントセッション	
LDAP 検索/秒	
LDAP 成功バインド/秒	
プロセッサ	% プロセッサ時間
セキュリティシステム全体の統計	Kerberos 認証
NTLM 認証	

追加リソース

- 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)
- Active Directory Domain Services のキャパシティプランニング (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で動作します。また、AWS マネージドアプリケーションとサービス 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 のために のコスト最適化手段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 SKUs が利用可能です。その他の SKUs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) 、リージョン間のデータ転送が含まれます。

コスト最適化に関する推奨事項

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、125 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 つのアカウント共有 (ソフト制限は約 30 個) をサポートし、マルチリージョンをサポートしていません。

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。

- 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 30 を超えるアカウントと共有されますか？
- Active Directory のオブジェクト数は 30,000 を超えていますか？

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 をデプロイしても、一方向のドアは通過しません。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。

Enterprise Edition でディレクトリ AWS Managed Microsoft AD を共有する場合、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、チェックしないとコストが急増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) と 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 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 AWS Managed Microsoft AD。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 に対して自己管理型の 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、 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 AWS Managed Microsoft AD 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、「」のドメインコントローラーのパフォーマンスメトリクス CloudWatchを確認してください。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圧倒され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圧倒されていると判断した場合、ドメインコントローラーを追加しても負荷は軽減されないため、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深く掘り下げる必要があります。すべてのドメインコントローラーが大量に使用されている場合、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 または SYSVOL 共有を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、これらの共有のコンテンツ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、「エンタープライズカスタマー」を参照してください。

追加リソース

- AWS Managed Microsoft AD クォータ (AWS Directory Service ドキュメント)
- AWS Directory Service 料金 (AWS ウェブサイト)
- での Active Directory ドメインサービス 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 、Amazon WorkSpaces、 QuickSightおよび 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 に簡単に接続できるプロキシサービスです。クラウドに情報をキャッシュする必要はありません。AD Connector を使用して、Active Directory に 1 つのサービスアカウントを追加できます。AD Connector を使用すると、ディレクトリの同期や、フェデレーションインフラストラクチャのホスティングにかかるコストと複雑さがなくなります。サービスの性質とその請求メカニズムにより、AD Connector のコスト最適化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 SKUs。これにより、サイズ設定が簡単になります。サイズ設定 SKUs には、スモールサイズとラージサイズの 2 つがあります。AD Connector に関連するコストの見積もりAWS Pricing Calculatorには、 を使用できます。

コスト最適化に関する推奨事項

バックエンドコンピューティングリソース以外には、コネクタのサイズが小さくても大きい場合でも違いはありません。

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は多数 (10,000 人以上) ですか？
- ユーザーは、ネストされた多数のグループ、ディープグループ、または循環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」と答えた場合は、大きなサイズを検討することをお勧めします。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な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 には関連していませんが、多くのユーザーが多くの、深い、または円形のネストされたグループのメンバーであるため、オンプレミスの Active Directory ドメインコントローラーは圧倒されています。

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、「エンタープライズカスタマー」を参照してください。

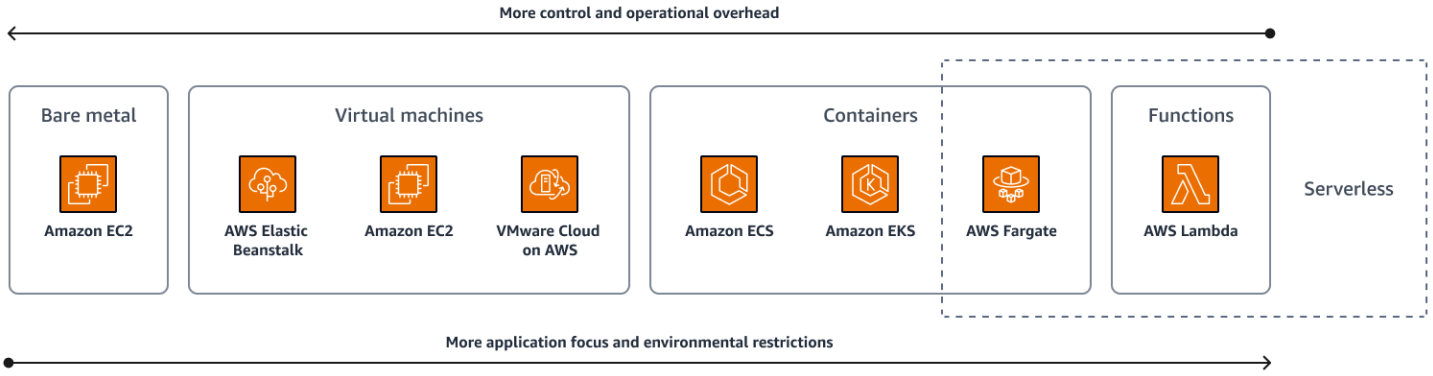
追加リソース

- 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)
- での Active Directory ドメインサービス AWS (AWS ホワイトペーパー)

.NET

の開発とデプロイ。NET アプリケーションは、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上で重要なキーです。多くのレガシー .NET アプリケーションでは、 でアプリケーションを実行するのに最も適したコンピューティング選択 AWS は、 AWS Elastic Beanstalk または Amazon Elastic Compute Cloud (Amazon) を介して仮想マシンを使用することですEC2。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、従来のコンピューティング選択のセットを使用し、 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 以降では、Graviton2 EC2ファミリーなどのARM64EC2インスタンスでワークロードの高性能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 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EC2。つまり、アプリケーションは、ビデオエンコーディング、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 () 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HPC。



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ために、 NETアプリケーションを調整するための推奨事項を提供します。

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 にリファクタリングし、NETLinux 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 の動的スケーリングをサポートします。NET フレームワークアプリケーション
- 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 を検討してください。NET
- 専用データベースを検討する

最新の にリファクタリングし、NETLinux に移行する

概要

レガシー をモダナイズします。NET フレームワークアプリケーションは、セキュリティ、パフォーマンス、スケーラビリティの向上に役立ちます。をモダナイズする効果的な方法。NET フレームワークアプリケーションは、最新の .NET バージョン (6+) に移行します。これらのアプリケーションをオープンソース に移行するための主な利点をいくつか示します。NET

- 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には
- 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

多くの組織が、古いバージョンの をまだ実行しています。NET フレームワーク。これは、古いバージョンの脆弱性が Microsoft によって対処されなくなったため、セキュリティリスクを引き起こす可能性があります。Microsoft は、 の最新バージョンのサポートを終了しました。NET フレームワーク 4.5.2、4.6、および 4.6.1。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、 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。NET

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) を考えてみましょう。これらのインスタンスは、ウェブサーバー、中規模データベース、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 (ライセンスを含む) に 4 GB vCPUs と 16 GB のメモリを搭載したオンデマンド m5.xlarge インスタンスの場合、月額 274.48 ドルかかります。Linux サーバー上の同じリソースは、毎月 140.16 ドルかかります。この例では、 からアプリケーションを移行するとコストが 49% 削減されます。NET の最新バージョンへのフレームワーク。NETLinux サーバーでアプリケーションを実行します。コストは、インスタンス を選択するときに選択したオプション (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) EC2 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 Pricing Calculator 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 の移植。NET アプリケーションを最新の にフレームワーク化します。NET には開発者の労力が必要です。アプリケーションとその依存関係を評価して、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、 をスキャンする補助ツールです。NET アプリケーションをフレームワーク化し、 を生成します。NET の互換性評価により、アプリケーションを Linux とより迅速に互換性を持つように移植できます。Porting Assistant for 。NET は、 との非互換性を識別しNET、既知の代替を見つけ、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、アプリケーションを Linux にモダナイズする際の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できるようになります。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 など、他のツールもあります。NET リファクタリングと 。NET Upgrade Assistant は、レガシー .NET フレームワークアプリケーションを最新の に移植するのに役立ちますNET。

コスト最適化に関する推奨事項

を移行するには。NET フレームワークアプリケーションでは、次の操作を行います。

1. 前提条件 – Porting Assistant for を使用するにはNET、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。マシン上のリソースには、1.8 以上の GHz処理速度、4 GB のメモリ、5 Gb のストレージスペースが必要です。詳細については、「 の移植アシスタント」の「前提条件」を参照してください。NET
2. 評価 – 実行可能ファイル (ダウンロード) ファイルとして の Porting Assistant をダウンロードしますNET。ツールをダウンロードしてマシンにインストールすると、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 と互換性APIsのない移植されたプロジェクト、パッケージ、 が含まれていますNET。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 」の「ソリューションの移植」を参照してください。NET
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は移植アシスタントによって変更されます。ログを確認して、ソースコードの変更を確認できます。ほとんどの場合、コードは本番稼働の準備を整えるために、移行とテストを完了する

ために追加の労力を必要とします。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 」の「ソリューションの移植」を参照してください。NET

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。をモダナイズするには、多くのビジネスおよび技術上の推進要因が考えられます。NET Linux コンテナにアプリケーションをフレームワークします。重要な要因の 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、総所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のクロスプラットフォームバージョン、 NETおよび コンテナに移行する際のライセンスコストを削減し、リソース使用率を最適化できます。

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EKSとしてAmazon ECSまたは Amazon 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

- VB NET上にアプリケーションが構築されており (2002 年のレガシーフレームワーク) 、それらを NETに移植する場合は、「レガシー VBNET アプリケーションの への移植」を参照してください。NET 6.0 with Porting Assistant for 。NET AWS ブログの Microsoft Workloads に投稿します。
- Windows Communication Foundation (WCF) にレガシーアプリケーションがあり、最新の で実行する場合はNET、 Core を採用できますWCF。詳細については、 の Porting Assistant を使用してレガシーWCFアプリケーションを CoreWCF にモダナイズするを参照してください。NET AWS ブログの Microsoft Workloads に投稿してください。
- 移植アシスタントを Visual Studio の拡張機能として追加できますIDE。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、「Accelerate .NET application modernization with Porting Assistant for 」を参照してください。NET ブログの Microsoft Workloads の Visual Studio IDE拡張機能の投稿。 AWS
- 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、デベロッパーが を使用および共有することを奨励できます。NET知識とベストプラクティスを移植します。
- AWS Toolkit for を使用して、Linux で .NET フレームワークアプリケーションを最新の .NET に移植できます。NET リファクタリング。詳細については、「Accelerate .NET modernization with

AWS Toolkit for 」を参照してください。NET ブログの Microsoft Workloads の記事のリファクタリング。 AWS

- のコンテナ化と移行を高速化できますASP。NET AWS を使用する へのコアアプリケーション AWS App2Container。

.NET アプリケーションをコンテナ化する

概要

コンテナは、一貫性と再現性のある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 AWS Fargate、サーバーレスコンテナサービスである を使用して、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しながら、.NET アプリケーションのコストを削減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、アプリケーション上のトラフィックと需要のレベルなどがあります。小規模またはシンプルなアプリケーションの場合、コンテナとそれに関連するサービスの管理のオーバーヘッドが実際にコストを増加させる可能性があるため、コンテナは従来のインフラストラクチャアプローチと比較して大幅なコスト削減を実現でき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、リソース使用率が向上し、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、次の点に注意してください。

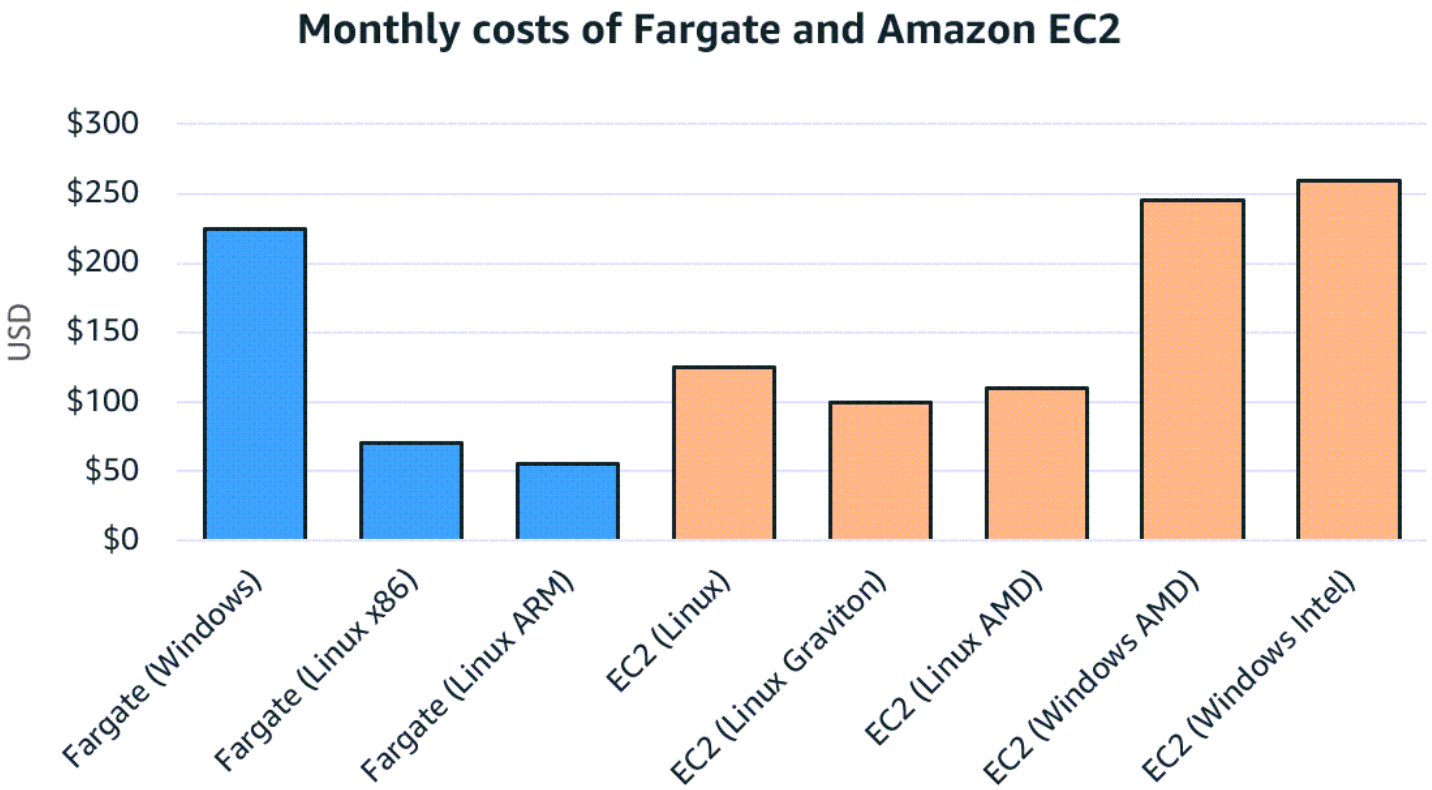
- アプリケーションのサイズと複雑さ – 大規模で複雑なアプリケーションは、より多くのリソースを必要とする傾向があり、リソース使用率の向上によりメリットが得られるため、コンテナ化に適しています。
- アプリケーションの数 – 組織がデプロイする必要があるアプリケーションが多いほど、コンテナ化によってコスト削減を実現できます。
- トラフィックと需要 – トラフィックと需要が多いアプリケーションは、コンテナが提供するスケーラビリティと弾力性の恩恵を受けることができます。これにより、コスト削減につながる可能性があります。

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、ライセンスに関する考慮事項により、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安くなるか、コストがかからなくなります。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 GB vCPUs と 8 GB のメモリが割り当てられ、それぞれ 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。

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選択して、 AWSEC2ベースのコンテナホストとサーバーレスまたは でコンテナを実行できますAWS Fargate。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、必要に応じてプレー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。

次の表は、Fargate と Amazon を使用した同等のコンテナの違いを示していますEC2。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、オフ時の使用率はゼロになります。ただし、Amazon ではECS、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容量を制御する必要があります。これにより、1 日 24 時間稼働するキャパシティが発生する可能性があり、最終的にコストが増大する可能性があります。



コスト最適化に関する推奨事項

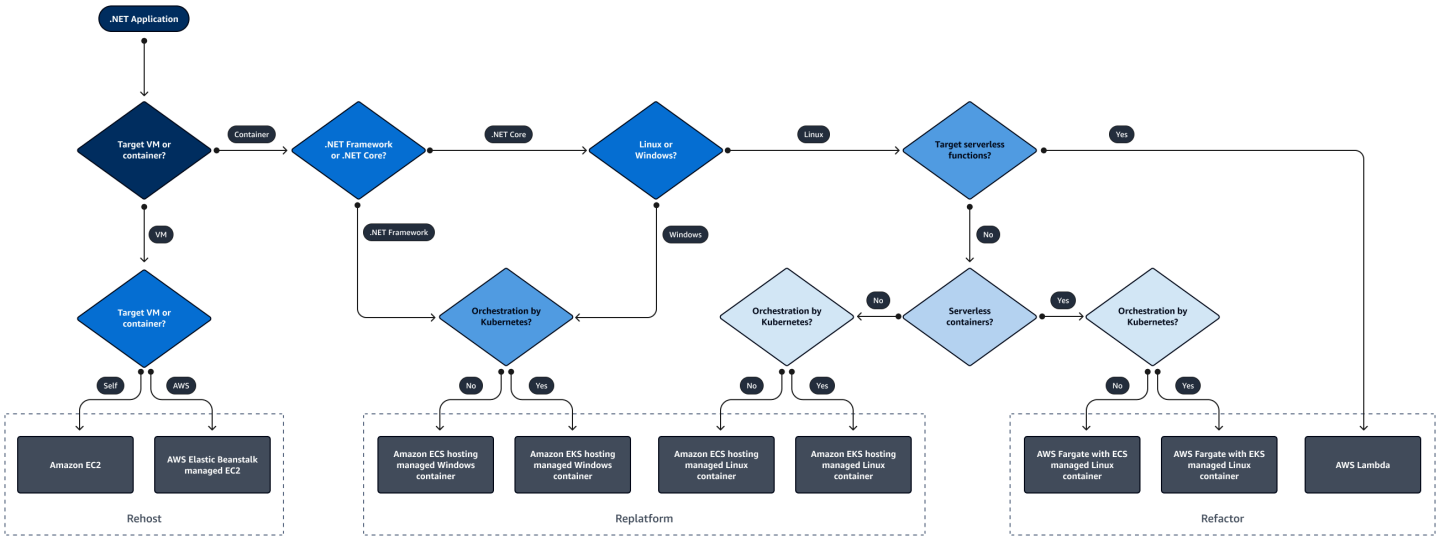
Windows ではなく Linux コンテナを使用する

Windows コンテナの代わりに Linux コンテナを使用すると、大幅な削減を実現できます。例えば、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。NET を実行する代わりに EC2 Linux 上の Core。NET EC2 Windows 上のフレームワーク。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 に対して Linux ベースのコンテナを実行する予定がある場合。NET フレームワークアプリケーションでは、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 に移植する必要があります (NET など。NET 6.0) を使用して Linux コンテナを使用します。主な考慮事項は、Linux コンテナのコスト削減と比較して、リファクタリングのコストを比較検討することです。アプリケーションを最新の に移植する方法の詳細についてはNET、 AWS ドキュメントの「Porting Assistant for NET」を参照してください。

最新の に移行するもう 1 つの利点 (NETつまり、 から遠ざかる。NET フレームワーク) は、追加のモダナイゼーション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性があり、コスト効率に優れたマイクロサービスベースのアーキテクチャに再設計することを検討できます。

次の図は、モダナイゼーションの機会を検討するための意思決定プロセスを示しています。



Savings Plans を活用する

コンテナは、Compute Savings Plansを活用して Fargate コストを削減するのに役立ちます。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

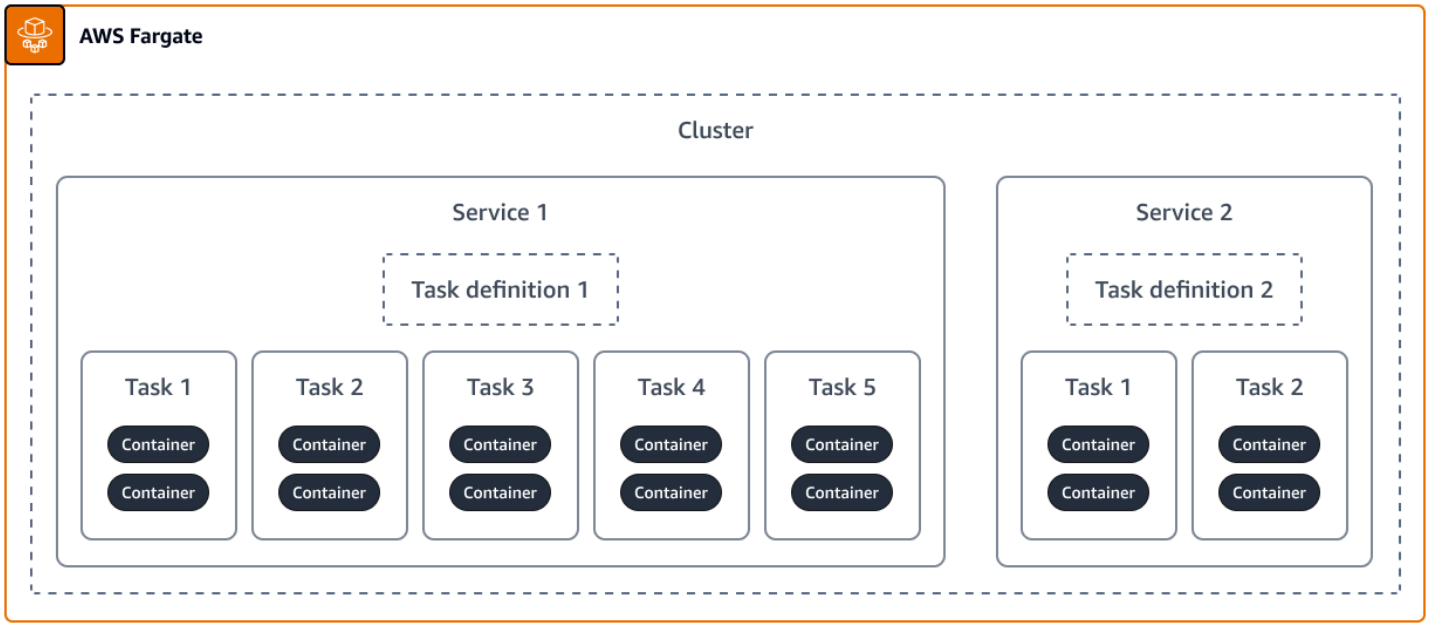
料金は、コンテナイメージのダウンロードを開始した時点から Amazon ECSタスクが終了するまで (最も近い秒に切り上げ) に使用される vCPU リソースとメモリリソースに基づいています。Savings Plans for Fargate は、1 年または 3 年の期間に特定の量のコンピューティング使用量 (1 時間あたりドルで測定) を使用するというコミットメントと引き換えに、Fargate の使用を最大 50% 削減します。AWS Cost Explorer を使用して、Savings Plan を選択するのに役立ちます。

Compute Savings Plans は、最も大きな節約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、 で t3.medium Linux インスタンスus-east-2と同じ Windows t3.medium インスタンスを実行している場合、Linux インスタンスは最初に Savings Plan のメリットを受け取ります。これは、Linux インスタンスには 50% のコスト削減の可能性があり、同じ Windows インスタンスには 35% のコスト削減の可能性があるためです。Amazon EC2や Lambda など AWS アカウント、 で実行中の他の Savings Plan の対象となるリソースがある場合、Savings Plan 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、このガイドの「 Savings Plans ドキュメント」および「Amazon での Windows の支出の最適化」セクションの「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」を参照してください。 Savings Plans EC2

適切なサイズの Fargate タスク

Fargate タスクのサイズが正しく設定され、コスト最適化が最大限に行われるようにすることが重要です。多くの場合、デベロッパーは、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、タスクのオーバー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、特定のタスク設定がさまざまな使用シナリオでどのように機能するかを理解することをお勧めします。負荷テスト結果、v CPU、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 が最適なタスクとコンテナサイズのレコメンデーションを生成する方法を示しています。



1 つのアプローチは、 の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負荷をシミュレートした後、ベースライン使用率に達するまでタスクの vCPU とメモリの設定を微調整できます。

追加リソース

- Amazon ECSと (Containers ブログ記事) のコスト最適化チェックリスト AWS FargateAWS
- 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)
- の移植アシスタント (NETAWS ドキュメント)
- (AWS ソリューションライブラリ) での分散負荷テスト AWS
- AWS Compute Optimizer 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、 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 で使用できる Graviton は 3 世代ありますEC2。このガイドでは、 での Graviton 2 および 3 の使用に焦点

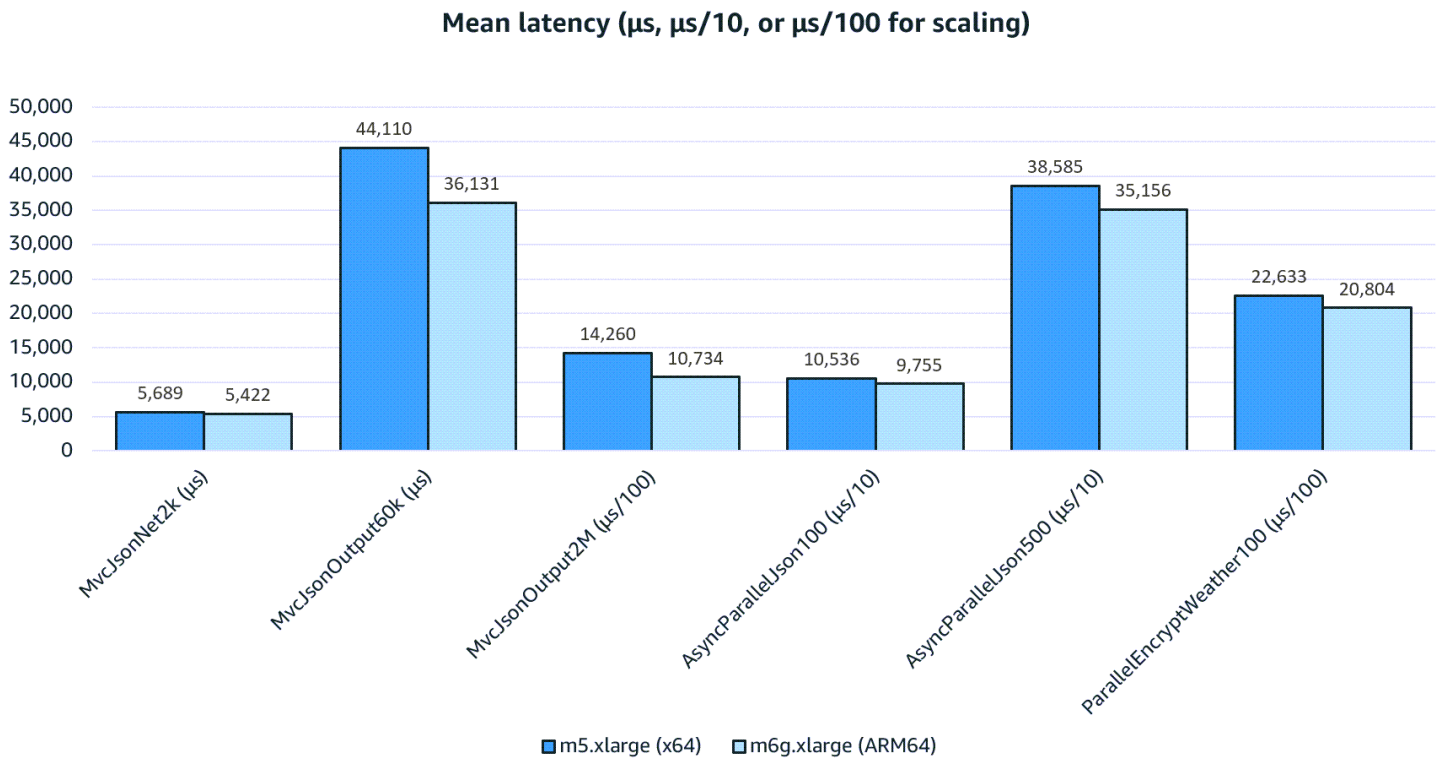
を当てています。NET最新バージョンの Graviton を使用すると大幅なコスト削減が見込めるためです。Graviton インスタンスは Linux オペレーティングシステムのみを実行することに注意してください。その結果、Graviton インスタンスは、Linux で実行される NETの強力なサービスですが、Windows オペレーティングシステムやレガシー のオプションではありません。NET フレームワークアプリケーション。

Graviton 3 は、同等のEC2インスタンスよりも 60% 効率が高く、パフォーマンスが最大 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% の節約を実現できます。レガシー を再ファクタリングした後。NET アプリケーションを最新の NETバージョンにフレームワークすると、Graviton インスタンスを使用する機能をアンロックできます。Graviton への移行は、. デベロッパーにとって効果的なコスト最適化手法ですNET。

次の表の例は、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。

効率が向上した理由の 1 つは、x86 と Graviton の間の 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 の結果は、同等の x86/x64 インスタンスよりも 40% の価格パフォーマンスが向上しています。Graviton3 では、Graviton2 に対して以下が提供されます。

- パフォーマンスが 25% 向上し、パフォーマンスプロファイルが向上
- 最大 2 倍の浮動小数点パフォーマンス
- 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内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のコスト削減の違いを示しています。

この表は、Graviton が 19.20% 削減されたことを示しています。

インスタンスタイプ	アーキテクチャ	vCPU	メモリ (GB)	時間あたりのコスト (オンデマンド)
t4g.xlarge	ARM	4	16	$0.1344
t3.xlarge	x86	4	16	$0.1664

この表は、Graviton が 14.99% 削減されたことを示しています。

インスタンスタイプ	アーキテクチャ	vCPU	メモリ (GB)	時間あたりのコスト (オンデマンド)
c7g.4xlarge	ARM	16	32	0.5781 ドル
c6i.4xlarge	x86	16	32	$0.6800

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、堅牢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。

コスト最適化に関する推奨事項

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について説明します。

Lambda でランタイム設定を変更する

の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。はNET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、「」のNET「Graviton」を参照してください GitHub。

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:

```
docker buildx build -t "myImageName:latest" --platform linux/amd64,linux/arm64 --push .
```

また、 などのツールを使用して AWS Cloud Development Kit (AWS CDK) 、ビルドのオーケストレーションに役立てることもできます。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 ARMから 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 を作成できますCPU。手順については、Microsoft ドキュメントの Arm64 およびその他のプラットフォームをターゲットにするようにプロジェクトを設定するには」を参照してください。

を使用している場合はNETCLI、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、Accelerate をご覧ください。NET の AWS Graviton2 で Arm64 を使用した 6 のパフォーマンス YouTube。依存関係の問題により、コンパイル時のエラーが発生し、個別に対処できます。依存関係にARMライブラリがある限り、移行は比較的簡単です。

追加リソース

- Amazon の Graviton ARMインスタンスとスポットインスタンスを使用してコンテナを構築し、保存する方法 ECS (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% のコストパフォーマンスを向上 (AWS ブログ)
- Arm ベースの AWS Graviton2 プロセッサへの AWS Lambda 関数の移行 (AWS ブログ)
- (AWS ブログ) を使用して . ウェブアプリケーションを構築して ARMにデプロイ AWS ECSします。NET AWS CDK
- Graviton Fast Start – ワークロードを AWS Graviton に移動するための新しいプログラム (AWS ブログ)
- AWS Graviton2 で .NET 5 を強化: ベンチマーク (AWS ブログ)

静的 の動的スケーリングをサポートします。NET フレームワークアプリケーション

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、柔軟性、または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能力です。これにより、ピーク使用量を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容量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よりも何倍も多くのトラフィックをすばやく取得できるサイバーマンデー (例えば、数分以内に何千パーセントも) は、弾力性の良い例です。

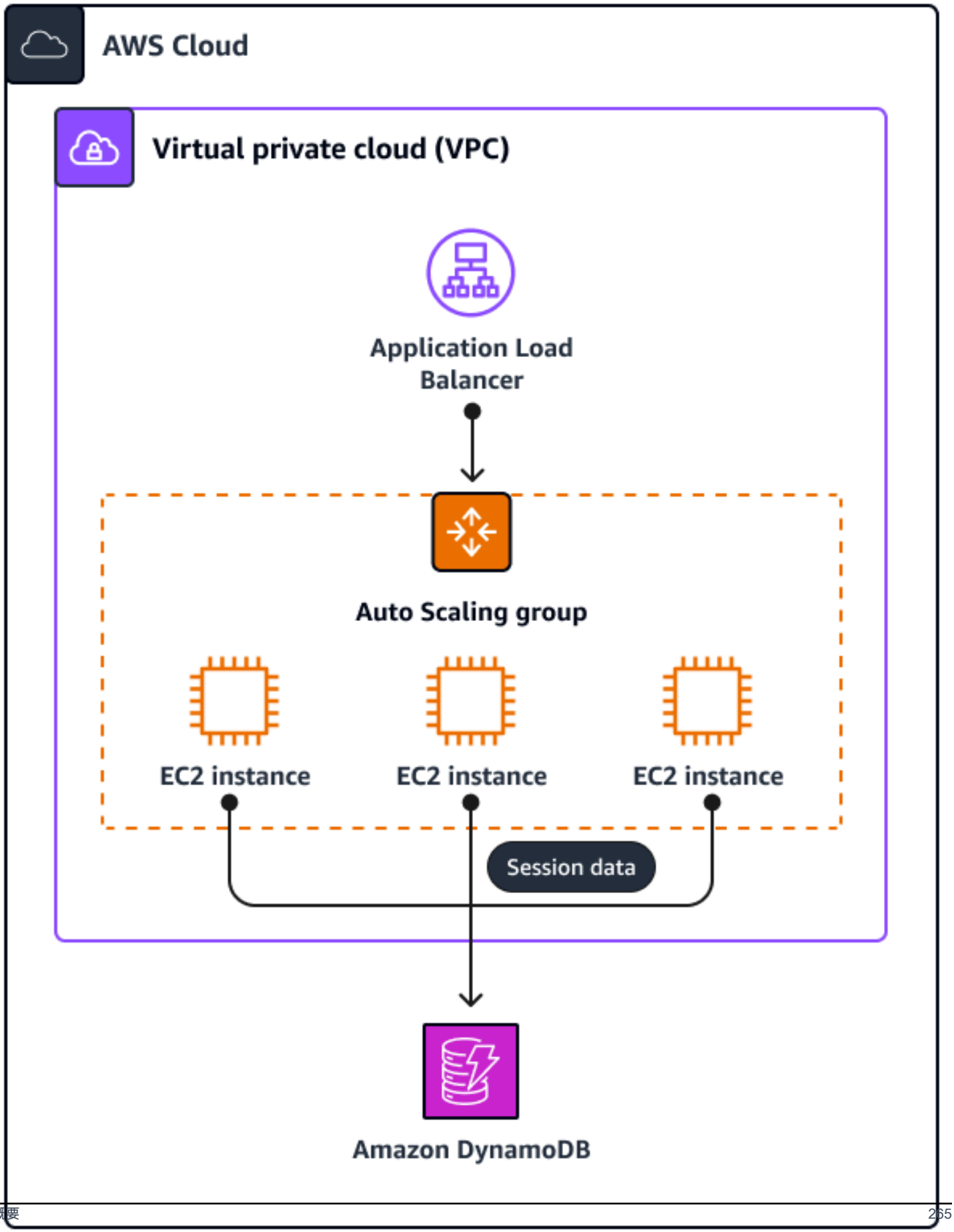
レガシー .NET ウェブアプリケーションをクラウドに持ち込む場合 (例: ASP。NET IIS) で実行されているフレームワークアプリケーションは、アプリケーションのステートフルな性質上、ロードバランシングされたサーバーファームをすばやくスケーリングすることは困難または不可能な場合があります。ユーザーセッションデータは、アプリケーションのメモリ内に保存されますASP。NET通常は、保持する必要があるクロスリクエストデータを保持するセッション状態または静的変数です。ユーザーセッションのアフィニティは通常、ロードバランサーのスティッキーセッションを通じて維持されます。

これは運用上困難であることが証明されています。容量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。これは遅いプロセスである可能性があります。パッチ適用時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。そのためには、ユーザーが再度ログインする必要があります。

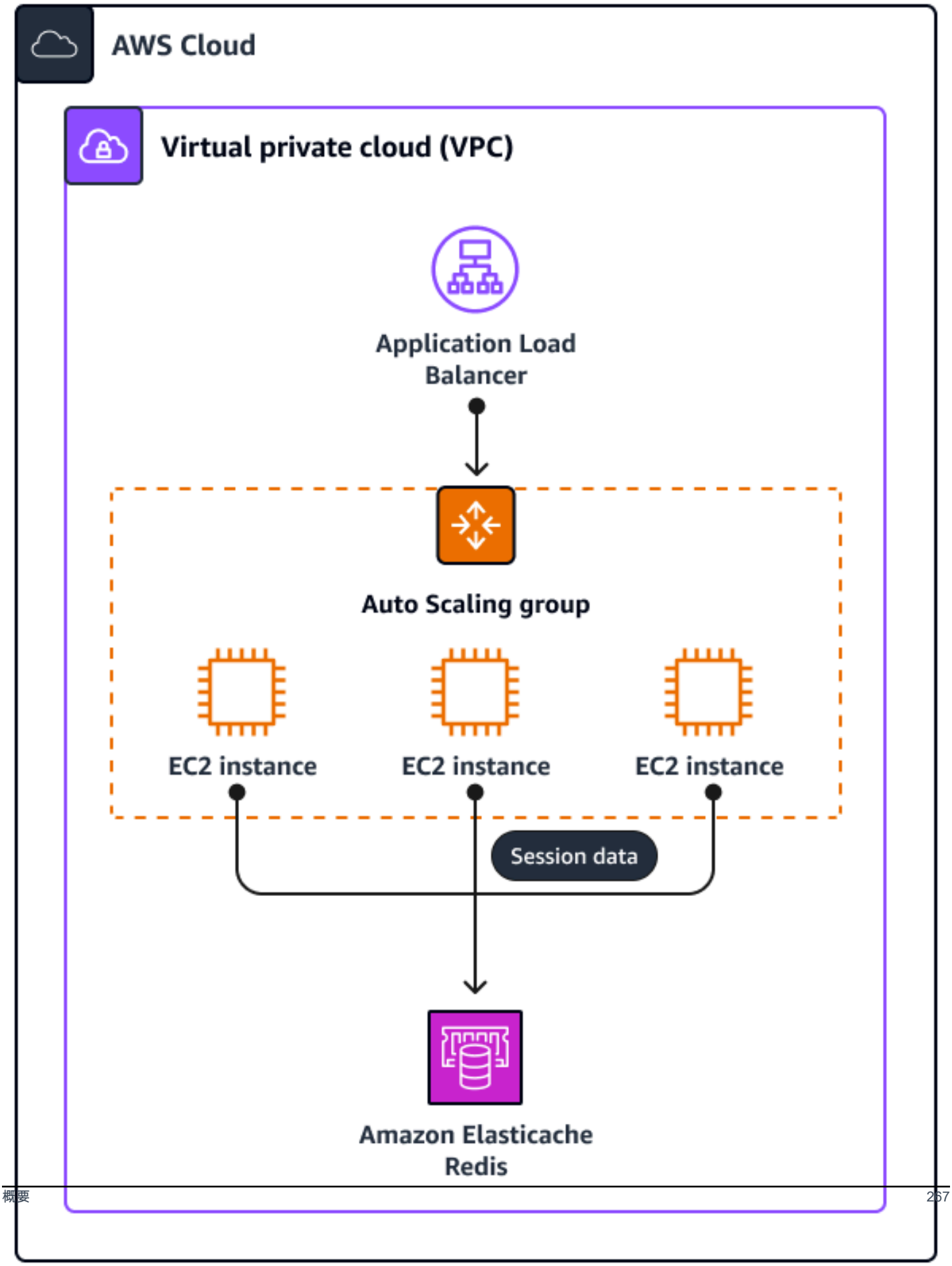
ASP.NET アプリケーションのセッション状態を一元化し、Auto Scaling ルールをレガシー ASP.NET アプリケーションに適用することで、クラウドの弾力性を活用し、アプリケーションの実行時にコスト削減を活用できます。例えば、コンピューティングのスケーラビリティによりコスト削減を実現できますが、リザーブドインスタンスの使用量の削減や Amazon スポットインスタンスの料金表の使用など、利用可能なさまざまな料金モデルから選択することもできます。

2 つの一般的な手法には、セッションステートプロバイダーとして Amazon DynamoDB を使用する方法とASP、.NET セッションストアとして Amazon ElastiCache (Redis OSS) を使用する方法があります。

次の図は、DynamoDB をセッション状態プロバイダーとして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、 ElastiCache (Redis OSS) をセッション状態プロバイダーとして使用するアーキテクチャを示しています。

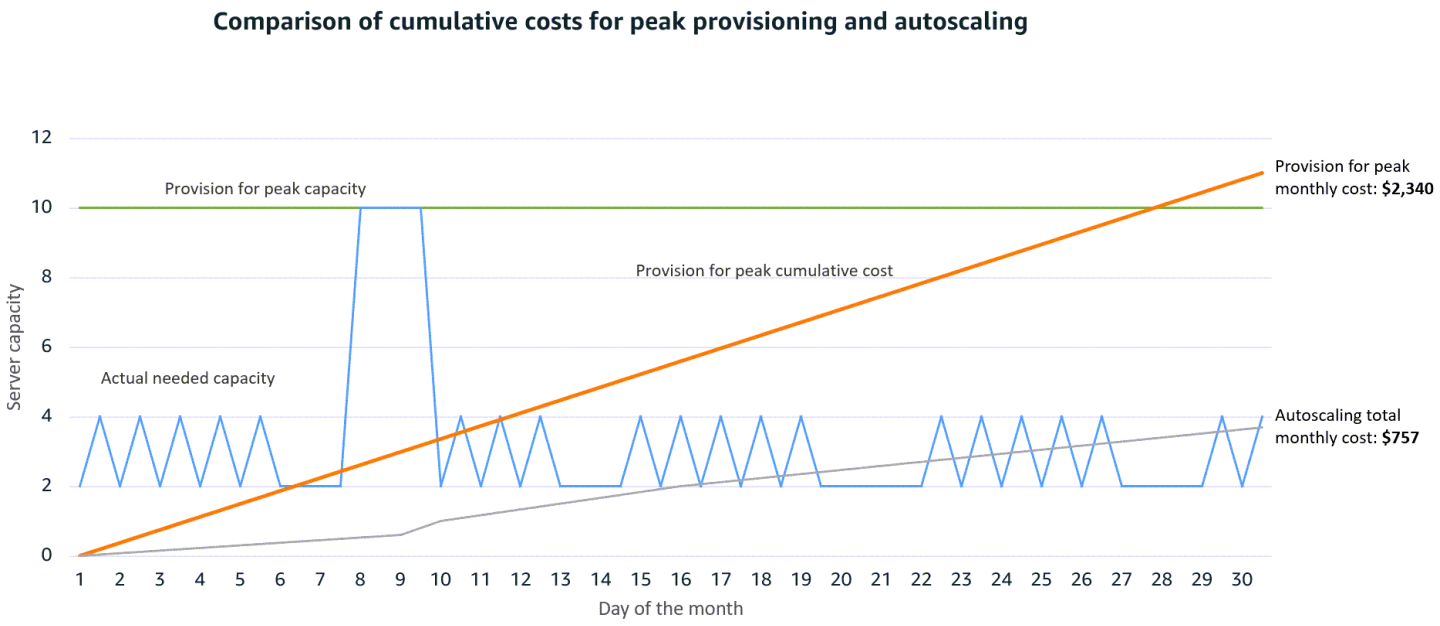

AWS Cloud
Virtual private cloud (VPC)
Application Load Balancer
Auto Scaling group
EC2 instance
EC2 instance
EC2 instance
Session data
Amazon Elasticache Redis

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、次の前提を行います。

- ローテーションから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変動は導入されません。
- 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線形にスケールされます (つまり、トラフィックの 2 倍には 2 倍のコンピューティングが必要です) 。
- 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 10 倍のトラフィックの 1 日分の異常なピーク (プロモーションセールなど) があります。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ベース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を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弾力性をどのように活用するかを示しています。これにより、約 68% の節約になります。



DynamoDB をセッション状態ストレージメカニズムとして使用する場合は、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料金は、1 か月あたり約 35.00 USD です。

(ElastiCache Redis OSS) をセッション状態ストレージメカニズムとして使用する場合は、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料金は、1 か月あたり約 91.00 USD です。

コスト最適化に関する推奨事項

最初のステップは、レガシー . NETアプリケーションでセッション状態を実装することです。をステートストレージメカニズム ElastiCache として使用する場合は、 AWS 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。DynamoDB を使用している場合は、「」のガイダンスに従ってくださいElastiCache ASP。NET デベロッパーツールブログのセッションストア。 AWS

アプリケーションがInProcセッションを使用して から開始する場合は、セッションに保存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デコレートします。例:

```
[Serializable()]
public class TestSimpleObject {
public string SessionProperty {get;set;}
}
```

さらに、 。NET は、使用中のすべてのサーバー間で同じMachineKeyである必要があります。これは通常、インスタンスが一般的な Amazon マシンイメージ () から作成された場合も同様ですAMI。例:

```
<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SystemWebSectionGroupMachineKey 「.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。これを実現するにはいくつかの方法があります。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。NET Auto Scaling グループのEC2インスタンスへのアプリケーションのフレームワーク：

- 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AMIを含む を設定します。その後、これを使用して Auto Scaling グループ の起動テンプレートAMIを設定できます。
- AWS CodeDeploy を使用して、Application. CodeDeploy enables と Amazon EC2 Auto Scaling との統合を直接デプロイします。これにより、アプリケーションのリリースAMIごとに新しい を作成する代わりに使用できます。

追加リソース

- EC2 Image Builder を使用してイメージを作成する (EC2 Image Builder ドキュメント)
- をデプロイしています。NET Visual Studio Team Services AWS CodeDeploy で を使用するウェブアプリケーション (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て、 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額に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。

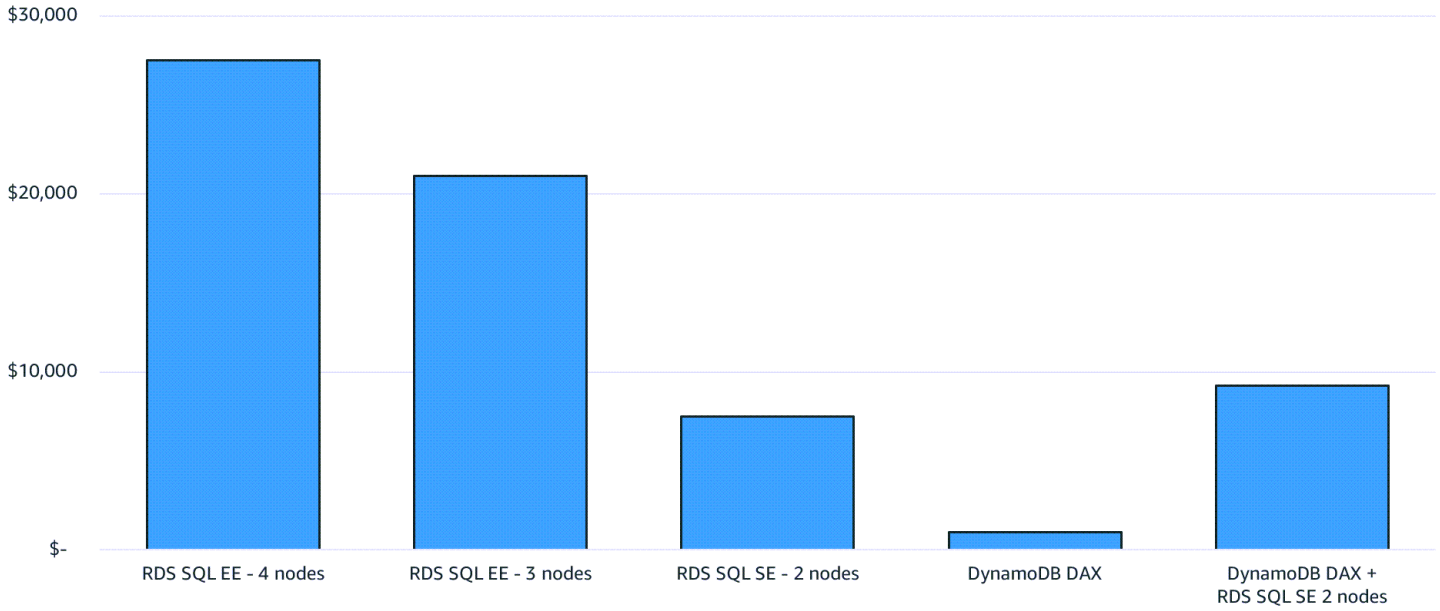
キャッシュは、SQLサーバーなどのより高価なリソースを使用する読み取り負荷の高いワークロードに関連するコストを削減するのに役立つ手法です。ワークロードに適した手法を使用することが重要です。例えば、ローカルキャッシュはスケーラブルではなく、アプリケーションのインスタンスごとにローカルキャッシュを維持する必要があります。潜在的なコストと比較してパフォーマンスへの影響を比較検討し、基盤となるデータソースのコストが低いほど、キャッシュメカニズムに関連する追加コストが相殺されます。

コストへの影響

SQL サーバーでは、データベースのサイズ変更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、ロード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ため、コストに影響を与える可能性があります。リードレプリカを使用している場合は、SQLServer Enterprise Edition でのみ利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、SQLServer Standard エディション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。Server Enterprise Edition を実行している 4 つの db.m4.2xlarge ノードSQLを持つ Amazon RDS for SQL Server を示します。これは、1 つのリードレプリカを持つマルチ AZ 設定にデプロイされます。排他的リードトラフィック (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 2 ノードの DynamoDB Accelerator (DAX) クラスターを使用します。

次の表は、高読み取りトラフィックを処理する専用リードレプリカが不要になった結果を示しています。



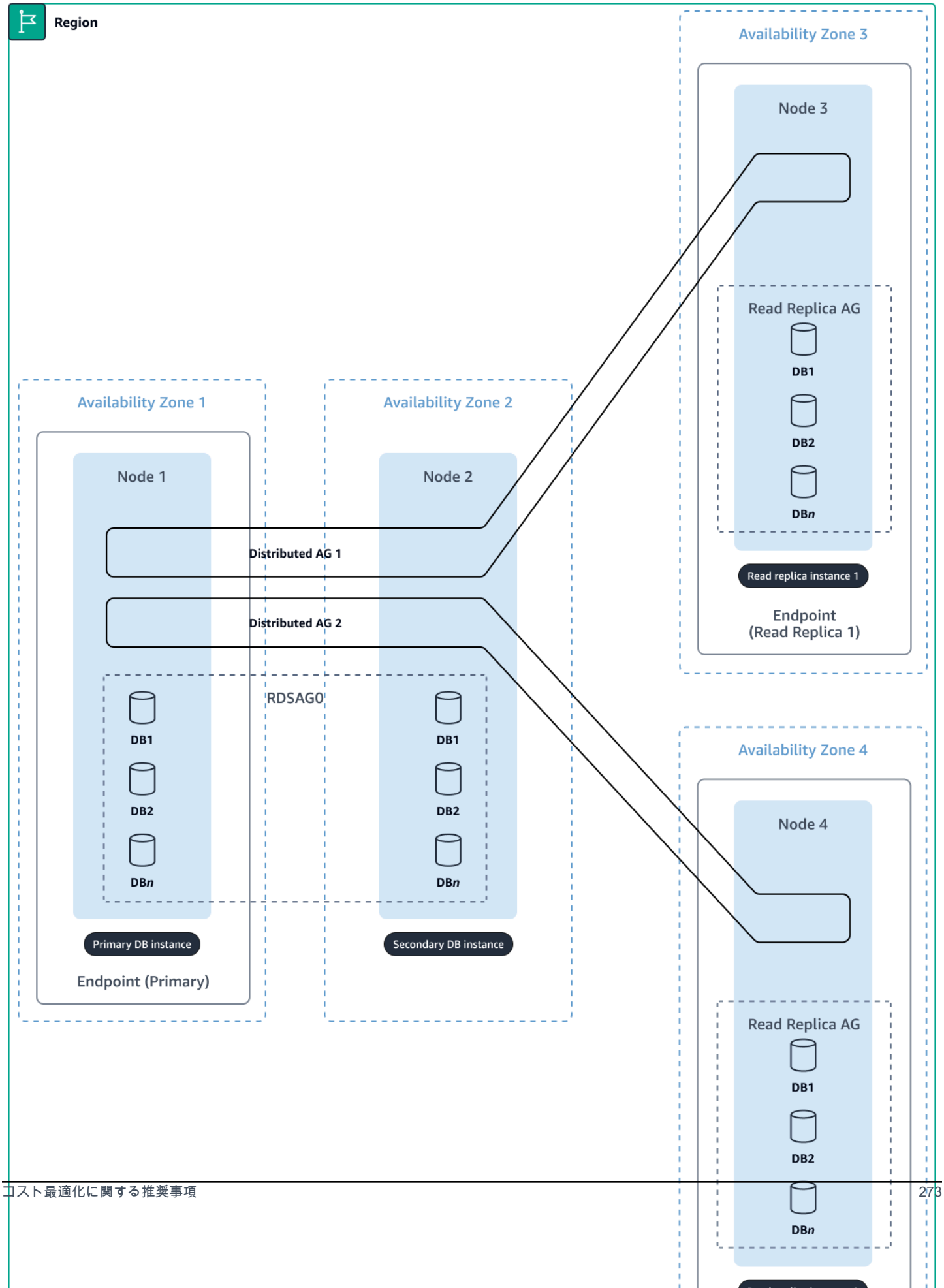
リードレプリカなしでローカルキャッシュを使用するか、Amazon 上の SQL Server をキャッシュレイヤーRDSとしてDAX並列に導入することで、大幅なコスト削減を実現できます。このレイヤーは SQL Server からオフロードされ、データベースの実行に必要な SQL Server のサイズが小さくなります。

コスト最適化に関する推奨事項

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の両方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最も一般的な方法の 1 つです。これは、実装が比較的簡単で直感的だからです。ローカルキャッシュには、データベースやその他のソースからコンテンツを取得し、メモリまたはディスクでローカルにキャッシュして、より迅速にアクセス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、アプリケーション状態やユーザー状態の保存など、キャッシュコンテンツを長期間保持する必要がある場合のユースケースなどです。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可用性SQLサーバー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。キャッシュにステートフル情報が格納されている場合、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、一部のインスタンスが過剰に利用される可能性があり、トラフィックの分散が不均等であるために十分に利用されないため、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。

. アプリケーションではNET、ローカルキャッシュをメモリ内またはローカルストレージのいずれかで使用できます。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# の SQL Server からのデータのローカルキャッシュインメモリとローカルストレージを実行するには、MemoryCache 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ますが、 LiteDBは高速で軽量な組み込みの NoSQL disk ベースのデータベースです。

インメモリキャッシュを実行するには、 を使用します。NET ライブラリ System.Runtime.MemoryCache。次のコード例は、System.Runtime.Caching.MemoryCache クラスを使用してメモリ内の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 などのより高価なリソースからデータを取得する必要性を減らすことで、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API。

コードの仕組みは次のとおりです。

1. MemoryCache という名前のプライベート静的インスタンス_memoryCacheが作成されます。キャッシュには、識別する名前 (dataCache) が付けられます。次に、キャッシュはデータを保存および取得します。
2. GetData メソッドは、stringキーと と呼ばれるFunc<T>委任という 2 つの引数を取る汎用メソッドですgetData。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ますが、getData委任は、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 を使用してデータを取得し_memoryCache.Get(key)、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代理を呼び出してデータを取得します。次に、 を使用してデータをキャッシュに追加します_memoryCache.Add(key, data,

DateTimeOffset.Now.AddMinutes(10))。この呼び出しでは、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ます。その時点で、データはキャッシュから自動的に削除されます。

5. ClearCache メソッドはstringキーを引数として受け取り、 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_memoryCache.Remove(key)。

```
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。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NET、「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。

を使用したキャッシュ DAX

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型の高可用性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 DynamoDB テーブルで読み取り

キャパシティーユニットをオーバープロビジョニングする必要性を減らすことで、 を使用してコストを削減できます。これは、読み取りが多く、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 はオンデマンドまたはプロビジョニングされた容量で料金設定されるため、1 か月あたりの読み取りと書き込みの回数がコストに寄与します。大量のワークロードを読み取る場合、DAX クラスターは DynamoDB テーブルの読み取り数を減らすことでコストを削減できます。を設定する方法についてはDAX、DynamoDB ドキュメントの「DynamoDB Accelerator (DAX) によるメモリ内アクセラレーション」を参照してください。 DynamoDB .NET アプリケーション統合の詳細については、「 DAXへの Amazon DynamoDB の統合」を参照してくださいASP。NET でのアプリケーション YouTube。

追加リソース

- DynamoDB Accelerator によるインメモリアクセラレーション (DAX) - Amazon DynamoDB (DynamoDB ドキュメント)DynamoDB
- Amazon DynamoDB DAXを に統合しますASP。NET アプリケーション (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 を検討してください。NET

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、サーバーレスアプローチが最新のアーキテクチャを構築するときに提供するスケーラビリティと俊敏性が原因です。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、デベロッパー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、 にとって特に魅力的なオプションです。NET は、インフラストラクチャコストの削減を検討しているデベロッパーにとって魅力的なオプションです。Lambda を使用すると、 NETデベロッパーは、スケーラビリティが高く、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と、デベロッパーはアプリケーションリクエストを処理するサーバーをプロビジョニングしなくなります。代わりに、デベロッパー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、潜在的にコスト効率が高くなります。その結果、アプリケーションが使用するリソースに対して

のみ支払いが行われ、リソースの使用率不足やサーバーメンテナンスコストを心配する必要はありません。

デベロッパーは、最新のクロスプラットフォーム .NET バージョンを使用して、高速、効率的、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 コア以降のバージョンは、以前の よりもサーバーレスプラットフォームでの実行に適した無料のオープンソースフレームワークです。NET フレームワークバージョン。これにより、デベロッパー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 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しています。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デベロッパー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を微調整し、Lambda 関数のメモリ割り当てのサイズを適正化し、ネイティブ AOTを使用して、Graviton ベースの関数に移動することで、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回数と各関数のメモリ量、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 40 万 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前後にあるワークロードの月額コストを大幅に削減できます。

また、Lambda 関数をターゲットとしてロードバランサーを使用する場合にも、追加コストが発生する場合があります。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るデータの量として計算されます。

コスト最適化に関する推奨事項

Lambda 関数の適切なサイズ

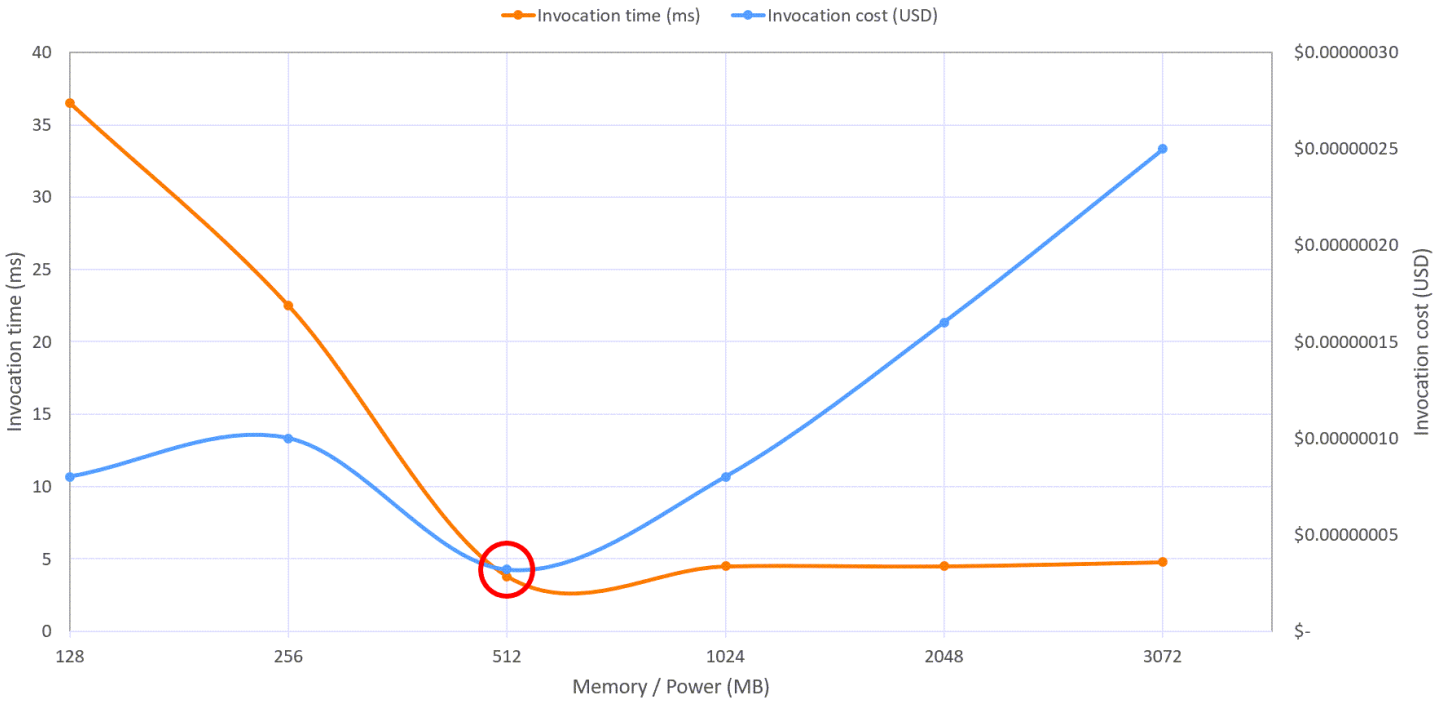
適切なサイジングは、. NETベースの Lambda 関数のコスト最適化に不可欠です。このプロセスでは、コードを変更することなく、パフォーマンスと費用対効果のバランスが取れた最適なメモリ設定を特定します。

Lambda 関数のメモリを 128 MB から 10,240 MB まで設定することで、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バインドされた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るため、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、. NETベースの Lambda 関数に最適な設定を特定することは、特に変更が頻繁に行われる場合、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ることなく、総呼び出し時間が向上し、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につれて、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、実行全体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であるためです。これは関数ごとに異なり、Lambda 関数でツールを使用して、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。



この演習は、新しい更新がリリースされたときに、統合テストの一環として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、定期的にこの演習を実行して、関数が調整され、適切なサイズであ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、プロセスに適切なサイズを追加できます。 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 ツールを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、「 でステートマシンを実行する方法」を参照してください GitHub。

Lambda はネイティブ もサポートしておりAOT、これにより . NETアプリケーションが事前にコンパイルされます。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「ネイティブAOTコンパイルによる関数NET」を参照してください。

アイドル状態の待機時間を避ける

Lambda 関数の期間は、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、または関数が他の関数のオーケストレーターとして機能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信システムなどのワークフローがある場合、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 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AWS Step Functionsとして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、ワークフローの合計期間ではなく、ワークフロー内の状態遷移ごとに請求されます。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します。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 ベースの関数に移動する

次世代 Graviton2 プロセッサを搭載した Lambda 関数が一般公開されました。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に対して、最大 19% のパフォーマンスを 20% 低コストで提供するように設計されています。レイテンシーが小さく、パフォーマンスが向上する Graviton2 プロセッサを搭載した 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パワーアップに最適です。

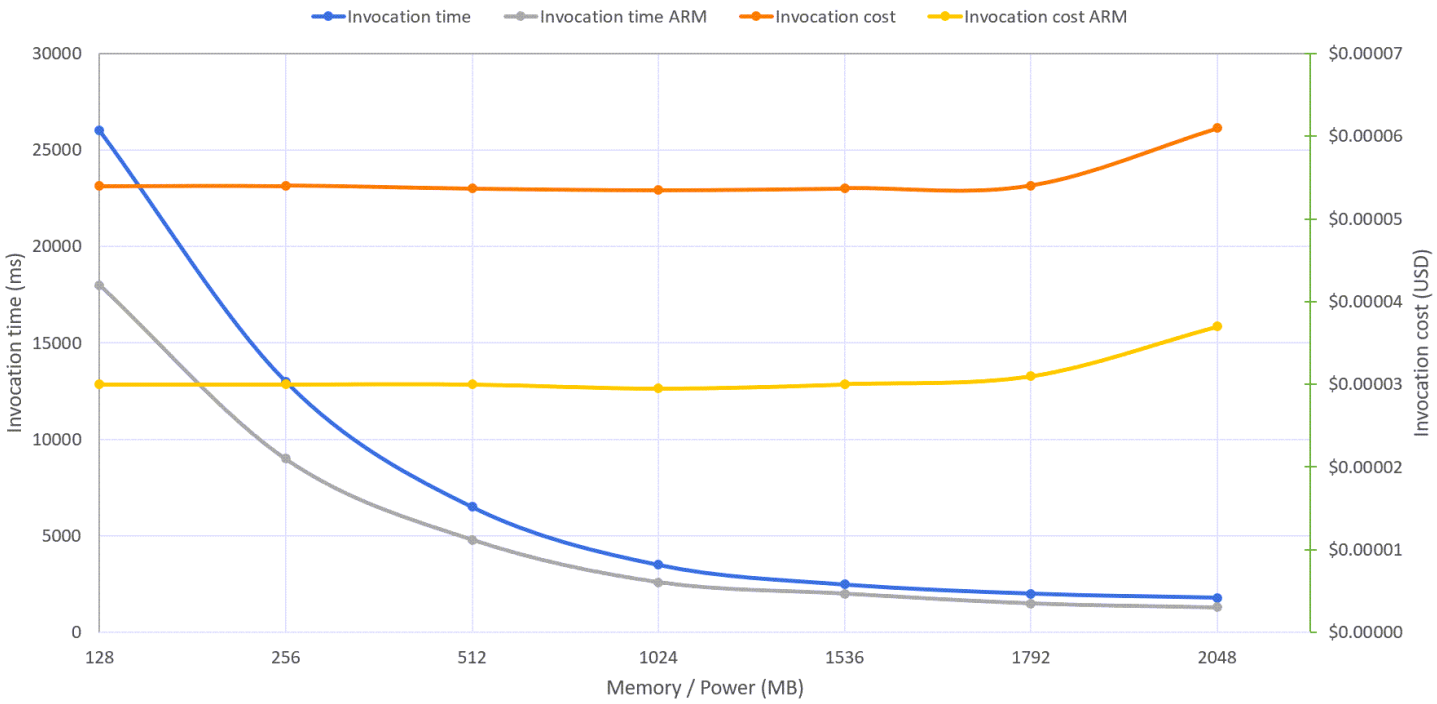
Graviton ベースの Lambda 関数への移行は、 にとってコスト効率の高いオプションです。NETLambda コストの最適化を検討しているデベロッパーにとっては、 です。Graviton ベースの関数は、従来の x86 プロセッサの代わりに ARMベースのプロセッサを使用します。これにより、パフォーマンスを犠牲にすることなく、大幅なコスト削減につながる可能性があります。

Graviton ベースの関数に移行するにはいくつかの利点がありますが、考慮すべき課題や考慮事項もあります。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。NETこれは、すべての 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の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。

を実行している場合。NET Lambda でアプリケーションをフレームワーク化し、サーバーレスを活用する場合は、 の Porting Assistant 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。 NETこれにより、レガシー .NET アプリケーションの最新の . への移植を加速しNET、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。



関数は単一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 1 v 以上にアクセスできますがCPU、この場合、関数は追加の電力を使用できません。同じ理由から、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、このコンピューティング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。

Graviton で および ARMベースのプロセッサを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 Management Console 、Lambda コンソール を開きます。
2. [Create function (関数の作成)] を選択します。

3. [関数名] に名前を入力します。
4. ランタイム の場合は、 を選択します。NET 6 (C#/PowerShell) 。
5. アーキテクチャ では、arm64 を選択します。
6. 必要な追加の設定を行い、関数の作成 を選択します。

追加リソース

- Lambda 関数をターゲットとして (AWS ドキュメント)
- (Compute Blog) を使用した AWS Lambda コストとパフォーマンスの最適化 AWS Compute OptimizerAWS
- 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)
- を使用してサーバーレス .NET AWS Lambda アプリケーションを構築する。NET 7 (AWS コンピューティングブログ)

専用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際の最もコストのかかる側面の 1 つは、SQLサーバーなどの商用データベースのライセンスです。多くの場合、企業はSQLサーバー上で任意のデータベースプラットフォームとして標準化され、組織の開発文化に深く根付いています。デベロッパーは通常、ユースケースに関係なくリレーショナルSQLサーバー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。

- ビジネスには、既にSQLサーバーインスタンスやライセンスが利用可能です。
- チームは、共有ライブラリ、、ORMsビジネスロジックを使用してSQLプログラミングモデルに慣れています。
- マネジメントは代替案を認識していません。
- デベロッパーは代替案を認識していません。

専用データベースは、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、より近代的なアーキテクチャ (マイクロサービスなど) を採用し、個々のアプリケーションの範囲が狭まるにつれて、企業によってますます採用されています。

専用に構築されたデータベースは、リレーショナルモデルを除外したり、いいSQLえ (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た場合、専用であると見なされます。専用データベースを使用すると、チームは アプリケーションNET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重労働の削減など、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の一貫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データベースがないSQL 非構造化データの高性能ストレージと取得 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	高性能 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データベースなしSQL 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や、他の遅いデー

データベース	タイプ	特性
		タストアの前での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 互換の専用データベース
Amazon Timestream	時系列	高スループットのデータ取り込みを時間順で行える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や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	指定された構造や他のデータとの関係が強制されることなくデータを保存するデータベースがないSQL 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よく使用されます

データベース	タイプ	特性
Amazon Neptune	グラフ	データおよびデータ項目間の接続の表現の両方を保持するデータベースがないSQL 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	ワイド列	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。

専用データベースの採用の大きな要因は、商用ライセンスの廃止に起因す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、ピーク使用量ではなく、平均ケースの容量をプロビジョニングできます。Timestream などの専用データベースはサーバーレスで、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、または行う意思がない場合、Babelfish for Aurora PostgreSQL を提供します。場合によっては、Babelfish では、変更をほとんど行わずに、既存のSQLサーバーアクセスコードを使用できます。

アプリケーション専用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のと同じ (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専用データベースに対処します。特定のアプリケーション (キャッシュなど) については、他の推奨事項で説明されています。

コストへの影響

専用データベースを にNET採用すると、ワークロードが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すると、コスト削減が副次的な目標になる可能性があります。

このガイドの範囲外では、アプリケーション専用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について説明します。詳細については、 AWS チュートリアルディレクトリの「専用データベース」を参照してください。

次の表は、SQLサーバーを専用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示しています。これらは単なる概算であることに注意してください。正確な本稼働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 の 100 GB SSD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専用データベースの見積もりですus-east-1。ライセンスコストには、SQLサーバー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/仕様	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	月額総コスト
SQL Amazon の Server Standard Edition EC2	ライセンスインクルード	r6i.2xlarge (8 CPU/64 GB RAM)	$1,345.36	$0.00	$1,345.36
SQL Amazon での Server Enterprise Edition EC2	ライセンスインクルード	r6i.2xlarge (8 CPU/64 GB RAM)	$2,834.56	$0.00	$2,834.56
SQL Amazon の Server	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	$644.56	$456.00	1,100.56 ドル

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/仕様	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	月額総コスト
Standard Edition EC2					
SQL Amazon での Server Enterprise Edition EC2	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	$644.56	$1,750.00	$2,394.56
SQL Amazon の Server Standard Edition RDS		db.r6i.2xlarge (8 CPU/64 GB RAM)	$2,318.30	$0.00	$2,318.30
SQL Amazon での Server Enterprise Edition RDS		db.r6i.2xlarge (8 CPU/64 GB RAM)	$3,750.56	$0.00	$3,750.56

次の表は、専用サンプル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/仕様	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	月額総コスト
Amazon Aurora PostgreSQL	r6g.2xlarge (8 CPU/64 GB RAM)	855.87 ドル	$0.00	855.87 ドル
DynamoDB	プロビジョンドベース 100 WCU/400 RCU	$72.00		$72.00

データベースエンジン	インスタンスタイプ/仕様	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	月額総コスト
Amazon DocumentDB	db.r6i.2xlarge (8 CPU/64 GB RAM)	$778.60		$778.60

> **Important**
>
> この表は、購入から 3 年間におけるソフトウェアアシュアランス付きSQLサーバー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、2 コアパック、3 年間。SQL Server Enterprise エディションの場合: $15,700、コアパック 2 個、3 年間。

専用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、専用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ています。このアーキテクチャスイッチを計画するときは、総所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、ワークロードごとに予想されるパフォーマンスと消費の綿密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、製品を維持すること end-of-support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の価値があります。

コスト最適化に関する推奨事項

サーバーNETにアクセスしている アプリケーションにはSQL、専用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サーバー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] (ライブラリ)	データベース	の置き換え	フレームワークの互換性
Npgsql エンティティフレームワークコアプロバイダー	Amazon Aurora PostgreSQL	エンティティフレームワークコアSQLサーバープロバイダー	最新 。NET
Npgsql エンティティフレームワーク 6 プロバイダー	Amazon Aurora PostgreSQL	Entity Framework 6.0 SQLサーバープロバイダー	。NET フレームワーク
Npgsql (ADO.NET 互換 PostgreSQL ライブラリ)	Amazon Aurora PostgreSQL	ADO.NET	。NET フレームワーク/最新 。NET
エンティティSQLフレームワークコアプロバイダー	Amazon Aurora MySQL	エンティティフレームワークコアSQLサーバープロバイダー	最新 。NET
ポメロEntity Framework Core。MySql	Amazon Aurora MySQL	エンティティフレームワークコアSQLサーバープロバイダー	最新 。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、すべてのコードは使用前に徹底的にテストする必要があります。

その他の専用データベースには、 にアクセスするためのライブラリがあります。NET互換性のあるライブラリを使用すると、専用データベースにアクセスできます。その例を以下に示します。

- 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 （ タイムストリームドキュメント)
- Cassandra の使用。NET プログラムで Amazon Keyspaces にアクセスするためのコアクライアントドライバー (Amazon Keyspaces ドキュメント)

- .NET を使用して Neptune DB インスタンスに接続する (Neptune ドキュメント)

専用構築データベースに移行する場合は、 から以下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サーバースキーマを 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サーバーから 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 Postgre とのSQLサーバーデータベースの互換性を確認するのに役立ちますSQL。

追加リソース

- SQL サーバーを Amazon Aurora Postgre に移行するためのガイダンスSQL (AWS データベースブログ)
- 。NET モダナイゼーションワークショップ (AWS Workshop Studio)
- Babelfish APP Modernization Immersion Day (AWS Workshop Studio)
- 。NET イマージョンデー (AWS Workshop Studio)
- での Amazon Timestream の開始方法。NET (GitHub)
- 最新の 用の専用データベース。NET アプリケーション 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 エキスパートにお問い合わせください。MACO の専門家が、質問への回答や懸念への対処に役立ちます。すでに 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 エキスパートに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 にお問い合わせください。
2. レコメンデーションを適用します。このガイドで学習した推奨事項、ベストプラクティス、戦略を適用し、MACO の専門家と話して学習した推奨事項、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 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 AWS Budgets 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。今後の更新について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 サーバーの更新	CPU SQL サーバーワークロードの最適化セクションを更新しました。	2024 年 10 月 25 日
SQL サーバーとコンテナの更新	Compute Optimizer 、サーバーワークロードのレコメンデーションの確認、および App2Container セクションを使用した Windows アプリケーションのリプラットフォームを使用して、サーバーSQLサイズの最適化を追加しました。 Trusted Advisor SQL App2Container	2024 年 6 月 29 日
SQL サーバーライセンスの最適化	Compute Optimizer セクションを使用して、Optimize SQL Server ライセンスを追加しました。	2024 年 5 月 22 日
初版発行	—	2023 年 12 月 21 日

AWS 規範ガイダンス用語集

以下は、 AWS 規範的ガイダンスが提供する戦略、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 Postgre SQL互換エディションに移行します。
- 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にある程度の最適化を導入します。例: オンプレミスの Oracle データベースを の Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行します AWS クラウド。
- 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 に移行します。
- 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: オンプレミスの Oracle データベースを のEC2インスタンスで Oracle に移行します AWS クラウド。
- 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。サーバーを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移行します。例: を移行する Microsoft Hyper-V アプリケーション 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。
- 使用停止 — 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

ABAC

属性ベースのアクセスコントロール を参照してください。

抽象化されたサービス

「 マネージドサービス」を参照してください。

ACID

原子性、一貫性、分離、耐久性 を参照してください。

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。より柔軟ですが、アクティブ/パッシブ移行 よりも多くの作業が必要です。

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の方法。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。

集計関数

行のグループで動作し、グループの単一の戻り値を計算SQLします。集計関数の例には、 SUMおよび が含まれますMAX。

AI

人工知能 を参照してください。

AIOps

人工知能オペレーション を参照してください。

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。

アプリケーションコントロール

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 の需要要素であり、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?」を参照してください。

人工知能オペレーション (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。移行戦略で AWS がどのようにAIOps使用されるかの詳細については、「 オペレーション統合ガイド」を参照してください。

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、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コントロール (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ABAC AWS「」の「」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

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安価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かつ効果的な計画を策定 AWS するのに役立つ 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まとめています。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に関する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

個人または組織に混乱または害を与えることを目的としたボット。

BCP

事業継続計画 を参照してください。

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 で動作グラフを使用して、失敗したログオン試行、疑わしいAPI呼び出し、および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Data in a behavior graph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。「endianness」も参照してください。

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

2 つの別々の環境を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は 1 つの環境 (青) で実行し、新しいアプリケーションバージョンは他の環境 (緑) 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インタラクション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をインデックス化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、個人や組織に混乱や害を与え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、ボットハーダーまたはボットオペレーター と呼ばれる 1 つ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、ボットとその影響をスケールするための最もよく知られているメカニズムです。

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、「ブランチについて (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 にアクセスするための簡単な手段を提供します。詳細については、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。AWS

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、ホワイトペーパー AWSでのコンテナ化されたマイクロサービスの実行 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。

事業継続計画 (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 Cloud Adoption Framework を参照してください。

Canary のデプロイ

エンドユーザーへのバージョンのスローリリースと増分リリース。自信が持てば、新しいバージョンをデプロイし、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

データキャプチャの変更を参照してください。

データキャプチャの変更 (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。は、同期を維持するために、ターゲットシステムの変更を監査したりレプリケートしたりするなど、CDCさまざまな目的で使用できます。

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します。AWS Fault Injection Service （ AWS FIS ） を使用して、 AWS ワークロードに負荷をかけ、そのレスポンスを評価する実験を実行できます。

CI/CD

継続的統合と継続的配信 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。

クライアント側の暗号化

ターゲットが AWS のサービス 受信する前に、データをローカルで暗号化します。

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テクノロジー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

組織は通常、 に移行するときに 4 つのフェーズを実行します AWS クラウド。

- 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
- 基盤 – クラウド導入を拡大するための基盤投資 (ランディングゾーンの作成、 の定義CCoE、オペレーションモデルの確立など)
- 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。 AWS 移行戦略との関連性については、「移行準備ガイド」を参照してください。

CMDB

設定管理データベース 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以下が含まれます。GitHub または Bitbucket Cloud。 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、デジタル画像や動画などのビジュアル形式から情報を分析および抽出する AI の分野。例えば、 はオンプレミスのカメラネットワークに CV を追加するデバイス AWS Panorama を提供し、Amazon SageMaker は CV の画像処理アルゴリズムを提供します。

設定ドリフト

ワークロードの場合、設定は想定された状態から変更されます。ワークロードが非準拠になる可能性があり、通常は段階的で意図しないものです。

設定管理データベース (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、移行のCMDBポートフォリオ検出および分析段階で のデータを使用します。

コンフォーマンスパック

コンプライアンス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、または組織全体に単一のエンティティとしてデプロイできます。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 is commonly described as a pipeline. CI/CD は、プロセスの自動化、生産性の向上、コード品質の向上、および迅速な提供に役立ちます。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

一元的な管理とガバナンスで分散された分散データ所有権を提供す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。信頼された ID のみが、期待されるネットワークから信頼された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、「 でデータ境界を構築する 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、通常、大量の履歴データが含まれ、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る。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。

ディープラーニング

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。

defense-in-depth

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 に採用する場合 AWS、リソースの保護に役立つように、 AWS Organizations 構造のさまざまなレイヤーに複数のコントロールを追加します。例えば、アプローチでは defense-in-depth、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。

デプロイメント

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、Implementing security controls on AWSのDetective controls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、リーンな製造プラクティス用に最初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ます。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 では、ファクトテーブル内の量的データに関するデータ属性を含む小さなテーブル。ディメンションテーブル属性は、通常、テキストフィールドまたはテキストのように動作する離散番号です。これらの属性は、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一般的に使用されます。

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)

災害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に使用する戦略とプロセス。詳細については、「 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。 AWS

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)。ストラングルラーの fig パターンでドメイン駆動型設計を使用する方法については、「従来の Microsoft のモダナイズ」を参照してくださいASP。NET (ASMX) コンテナと Amazon API Gateway を使用してウェブサービスを段階的に更新する「」。

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。例えば、 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 、ガバナンス要件のコンプライアンスに影響を与える可能性のあるランディングゾーンの変化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。

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、レスポンスタイムを向上させることができます。

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プロセス。

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) 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、インターフェイスエンドポイントを作成することでVPC、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、アイデンティティとアクセスの管理、検出コントロール、インフラストラクチャのセキュリティ、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、サマリー統計を計算し、データ視覚化を作成することで実行されます。

F

ファクトテーブル

星スキーマ の中央テーブル。ビジネスオペレーションに関する定量的なデータを保存します。通常、ファクトテーブルには 2 つのタイプの列が含まれます。つまり、メジャーを含む列と、ディメンションテーブルへの外部キーを含む列です。

フェイルファースト

開発ライフサイクルを短縮するために頻繁で増分的なテストを使用する哲学。これはアジャイルアプローチの重要な部分です。

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が、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 Explanations (SHAP) や統合勾配など、さまざまな手法で計算できる数値スコアとして表されます。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。

FGAC

「きめ細かなアクセスコントロール」を参照してください。

きめ細かなアクセスコントロール (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

段階的なアプローチを使用する代わりに、変更データキャプチャによる継続的なデータレプリケーションを使用して、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。

G

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)

Amazon では CloudFront、特定の国のユーザーがコンテンツディストリビューションにアクセスできないようにするオプションです。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分散の制限」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 ワークフローはレガシーと見なされ、トランクベースのワークフローはモダンで望ましいアプローチです。

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

組織全体のリソース、ポリシー、コンプライアンスを管理するのに役立つ大まかなルール (OUs) 。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、 AWS Security Hub、Amazon GuardDuty、 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。

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。

ハイアベイラビリティ (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。

同種データベースの移行

ソースデータベースを、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に移行する (Microsoft SQL Server から Amazon RDS for SQL Server など) 。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。緊急性のため、通常、ホットフィックスは一般的な DevOps リリースワークフローの外部で行われます。

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 日間の平均使用量CPUとメモリ使用量が 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

「産業用モノのインターネット」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

既存のインフラストラクチャを更新、パッチ適用、または変更する代わりに、本番稼働ワークロード用に新しいインフラストラクチャをデプロイするモデル。イミュータブルインフラストラクチャは、本質的に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 Framework」の「イミュータブルインフラストラクチャのベストプラクティスを使用したデプロイ」を参照してください。

インバウンド (イングレス) VPC

AWS マルチアカウントアーキテクチャでは、 VPCが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、ルーティングします。AWS セキュリティリファレンスアーキテクチャでは、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インターフェイスVPCsを保護するために、インバウンド、アウトバウンド、および検査でネットワークアカウントを設定することをお勧めします。

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

接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩を通じて、製造プロセスのモダナイゼーションを指すために 2016 年に Klaus Schwab によって導入された用語。

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。

産業用モノのインターネット (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、「産業用モノのインターネット (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。

検査 VPC

AWS マルチアカウントアーキテクチャでは、 VPCs (同一または異なる 内の 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VPCを管理する一元化された です。AWS セキュリティリファレンスアーキテクチャでは、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インターフェイスVPCsを保護するために、インバウンド、アウトバウンド、および検査でネットワークアカウントを設定することをお勧めします。

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

「モノのインターネット」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 の基盤を提供しますITSM。

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

ラベルベースのアクセスコントロール (LBAC)

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ている必須のアクセスコントロール (MAC) の実装。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。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースのアクセスコントロール」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「最小権限の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。「endianness」も参照してください。

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または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を混乱させたり、機密情報を漏洩したり、不正アクセスを受けたりする可能性があります。マルウェアの例としては、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 は、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 AWS を操作し、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。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、マネージドサービスの例です。これらは抽象サービス とも呼ばれます。

製造実行システム (MES)

原材料を作業現場の最終製品に変換する生産プロセスを追跡、モニタリング、文書化、制御するためのソフトウェアシステム。

MAP

「移行促進プログラム」を参照してください。

メカニズム

ツールを作成し、ツールの採用を推進し、調整を行うために結果を検査する完全なプロセス。メカニズムは、動作時にそれ自体を強化して改善するサイクルです。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

の組織の一部である管理アカウント AWS アカウント を除くすべての 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。

メッセージキューイングテレメトリトランスポート (MQTT)

リソースに制約のある IoT デバイス向けの、 machine-to-machineパブリッシュ/サブスクライブパターンに基づく軽量 (M2M) 通信プロトコル。

マイクロサービス

明確に定義された上で通信APIsし、通常、小規模な自己完結型チームが所有する、小規模で独立したサービス。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 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APIs。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。

移行促進プログラム (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、従来の移行を系統的な方法で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して高速化するための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、オペレーション、ビジネスアナリストと所有者、移行エンジニア、デベロッパー、スプリントに携わる 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には、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 EC2を使用して Amazon への移行をリホストします。

移行ポートフォリオ評価 (MPA)

に移行するためのビジネスケースを検証するための情報を提供するオンラインツール AWS クラウド。MPA は、詳細なポートフォリオ評価 (サーバーの適正サイズ、料金、TCO比較、移行コス

ト分析) と移行計画 (アプリケーションデータ分析とデータ収集、アプリケーショングループ化、移行の優先順位付け、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)

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 AWS CAF。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 の最初のフェーズです。

移行戦略

ワークロードを に移行するために使用するアプローチ AWS クラウド。詳細については、この用語集の「7 Rs エントリ」および「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、「」の「アプリケーションのモダナイズ戦略 AWS クラウド」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、「」の「アプリケーションのモダナイゼーション準備状況の評価 AWS クラウド」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

「移行ポートフォリオ評価」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。

ミュータブルインフラストラクチャ

本番ワークロードの既存のインフラストラクチャを更新および変更するモデル。Well-Architected Framework AWS 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をベストプラクティスとして使用することを推奨しています。

O

OAC

「オリジンアクセスコントロール」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

「運用レベルの契約」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 - 統合アーキテクチャ (OPC-UA)

産業オートメーション用の (M2M) machine-to-machine通信プロトコル。OPC-UA は、データの暗号化、認証、認可スキームとの相互運用性標準を提供します。

運用レベルの契約 (OLA)

サービスレベルの契約 () をサポートするために、IT グループが相互にどのような機能を提供することを約束するかを明確にする契約SLA。

運用準備状況のレビュー (ORR)

インシデントや潜在的な障害の範囲を理解、評価、防止、または軽減するのに役立つ質問とそれに関連するベストプラクティスのチェックリスト。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)

物理環境と連携して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ハードウェアおよびソフトウェアシステム。製造では、OT と情報技術 (IT) システムの統合が、Industry 4.0 変換の主要な焦点です。

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 を口グに記録する 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、 ドキュメントの「組織の証跡の作成」を参照してください。 CloudTrail

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、変化の採用を加速し、移行に伴う問題に対処し、文化的および組織的な変化を推進することで、組織が新しいシステムや戦略の準備と移行を支援します。 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から、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。

オリジンアクセスコントロール (OAC)

では CloudFront、Amazon Simple Storage Service (Amazon S3) コンテンツを保護するためのアクセスを制限するための拡張オプションです。OAC は、すべての の S3 バケット AWS リージョン、 AWS KMS （ SSE-KMS) によるサーバー側の暗号化、および 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)

では CloudFront、Amazon S3 コンテンツを保護するためにアクセスを制限するオプションがあります。を使用する場合OAI、 は Amazon S3 が認証できるプリンシパル CloudFront を作成します。認証されたプリンシパルは、特定の CloudFront ディストリビューションを介してのみ S3 バケット内のコンテンツにアクセスできます。も参照してください。これによりOAC、より詳細で拡張されたアクセスコントロールが提供されます。

ORR

「運用準備状況レビュー」を参照してください。

OT

運用技術 を参照してください。

アウトバウンド (出力) 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VPCを処理する です。AWS セキュリティリファレンスアーキテクチャでは、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インターフェイスVPCsを保護するために、インバウンド、アウトバウンド、および検査でネットワークアカウントを設定することをお勧めします。

P

アクセス許可の境界

ユーザーまたはロールが持つことができる最大アクセス許可を設定するために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。例としてPIIは、名前、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

「プログラム可能な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

アクセス許可の定義 (アイデンティティベースのポリシーを参照)、アクセス条件の指定 (リソースベースのポリシーを参照)、または の組織内のすべてのアカウントに対する最大アクセス許可の定義 AWS Organizations （ サービスコントロールポリシーを参照) が可能なオブジェクト。

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。詳細については、マイクロサービスでのデータ永続性の有効化 を参照してください。

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、「移行準備状況ガイド」を参照してください。

述語

`true` または を返すクエリ条件。一般的には `false`WHERE句にあります。

述語プッシュダウン

転送前にクエリ内のデータをフィルタリングするデータベースクエリ最適化手法。これにより、リレーショナルデータベースから取得して処理する必要があるデータの量が減少し、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、Implementing security controls on AWSのPreventative controls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ロール AWS アカウント、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「ロールの用語と概念」を参照してください。 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html#id_roles_terms-and-concepts

プライバシーバイデザイン

エンジニアリングプロセス全体を通してプライバシーを考慮に入れた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

Amazon Route 53 が 1 つ以上の 内のドメインとそのサブドメインのDNSクエリに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VPCs。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

非準拠のリソースのデプロイを防ぐ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、プロビジョニングされる前にリソースをスキャンします。リソースがコントロールに準拠していない場合、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「 での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)

設計、開発、発売から成長と成熟、減少と削除に至るまで、ライフサイクル全体にわたる製品のデータとプロセスの管理。

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)

製造では、マシンをモニタリングし、製造プロセスを自動化する、信頼性が高く適応性の高いコンピュータです。

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。

publish/subscribe (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で、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 ではMES、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できるチャネルにイベントメッセージを公開できます。システムは、発行サービスを変更せずに新しいマイクロサービスを追加できます。

Q

クエリプラン

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する手順などの一連のステップ。

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

責任、説明責任、相談、情報 (RACI) 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

責任、説明責任、相談、情報 (RACI) を参照してください。

RCAC

行と列のアクセスコントロール 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。

再設計

7 Rs を参照してください。

復旧ポイントの目的 (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 とは分離され、独立しています。詳細については、AWS リージョン 「 を使用できるアカウントを指定する」を参照してください。

回帰

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。

プラットフォーム変更

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

中断に抵抗または回復するアプリケーションの機能。高可用性とディザスタリカバリは、 で障害耐性を計画する際の一般的な考慮事項です AWS クラウド。詳細については、AWS クラウド「レジリエンス」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。

責任、説明責任、相談、情報 (RACI) 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。サポートを含めると、マトリックスはRASCIマトリックス と呼ばれ、除外すると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、Implementing security controls on AWSのResponsive controls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。

ローテーション

攻撃者が認証情報にアクセスすることをより困難にするために、シークレットを定期的に更新するプロセス。

行と列のアクセスコントロール (RCAC)

アクセスルールが定義されている基本的で柔軟なSQL式の使用。RCAC は、行アクセス許可と列マスクで構成されます。

RPO

「復旧ポイントの目的」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 (IdPs) が使用す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) AWS Management Console が有効になるため、ユーザーは にログインしたり、 AWS API組織内のすべてのユーザーIAMに対して でユーザーを作成したりすることなく 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「2.0 SAML 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

「監視コントロールとデータ取得」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

では AWS Secrets Manager、暗号化された形式で保存する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。シークレット値は、バイナリ、1 つの文字列、または複数の文字列にすることができます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの内容」を参照してください。

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、予防的 、検出的 、応答的 ???、およびプロアクティブ の 4 つの主なタイプがあります。

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。

セキュリティ情報とイベント管理 (SIEM) システム

セキュリティ情報管理 (SIM) システムとセキュリティイベント管理 (SEM) 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、その他の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、アラートを生成します。

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に自動的に応答または修正するように設計された、事前定義された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としては、VPCセキュリティグループの変更、Amazon EC2インスタンスのパッチ適用、認証情報のローテーションなどがあります。

サーバー側の暗号化

送信先で、それ AWS のサービス を受信する 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。SCPs ガードレールを定義するか、管理者がユーザーまたはロールに委任できるアクションの制限を設定します。を許可リストまたは拒否リストSCPsとして を使用して、許可または禁止されるサービスまたはアクションを指定できます。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URLの 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。

サービスレベル契約 (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、スループットなど、サービスのパフォーマンス側面の測定。

サービスレベルの目標 (SLO)

サービスレベルインジケータ によって測定される、サービスの正常性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 について共有する責任を説明するモデル。 クラウドのセキュリティ AWS はクラウドのセキュリティに責任があり、クラウドのセキュリティはユーザーの責任です。詳細については、責任共有モデルを参照してください。

SIEM

セキュリティ情報とイベント管理システム を参照してください。

単一障害点 (SPOF)

システムを混乱させる可能性のある、アプリケーションの単一の重要なコンポーネントの障害。

SLA

「サービスレベル契約」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。

split-and-seed 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、「」の「アプリケーションのモダナイズに対する段階的なアプローチ AWS クラウド」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

1 つの大きなファクトテーブルを使用してトランザクションデータまたは測定データを保存し、1 つ以上の小さなディメンションテーブルを使用してデータ属性を保存するデータベース組織構造。この構造は、データウェアハウスまたはビジネスインテリジェンス目的で使用するため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。このパターンを適用する方法の例については、「従来の Microsoft のモダナイズ」を参照してくださいASP。NET (ASMX) コンテナと Amazon API Gateway を使用してウェブサービスを段階的に更新する「」を参照してください。

サブネット

内の IP アドレスの範囲VPC。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。

監視コントロールとデータ取得 (SCADA)

製造では、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稼働をモニタリングするシステムです。

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

ユーザーインタラクションをシミュレートして潜在的な問題を検出したり、パフォーマンスをモニタリングしたりする方法でシステムをテストします。Amazon CloudWatch Synthetics を使用して、これらのテストを作成できます。

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。

トランジットゲートウェイ

VPCs とオンプレミスの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トランジットハブ。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

組織内でタスクを実行するために指定したサービスに、ユーザーに代わってそのアカウント AWS Organizations で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 つのピザを食べることができる小さな 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。詳細については、深層学習システムにおける不確実性の定量化 ガイドを参照してください。

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VPCsできる 2 つの間の接続。詳細については、Amazon VPCドキュメントのVPC「ピアリングとは」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する行のグループに対して計算を実行するSQL関数。ウィンドウ関数は、移動平均の計算や、現在の行の相対位置に基づく行の値へのアクセスなどの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

「書き込み」を 1 回参照し、多くの を読み取ります。

WQF

AWS 「ワークロード認定フレームワーク」を参照してください。

1 回書き込み、多数読み込む (WORM)

データを 1 回書き込み、データの削除や変更を防ぐストレージモデル。許可されたユーザーは、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 を利用する攻撃、通常はマルウェア。

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。

ゾンビアプリケーション

平均使用量CPUとメモリ使用量が 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。