



Amazon Timestream for InfluxDB 用の AWS Well-Architected フレームワーク
の適用 InfluxDB

AWS 規範ガイドンス



AWS 規範ガイド: Amazon Timestream for InfluxDB 用の AWS Well-Architected フレームワークの適用 InfluxDB

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

序章	1
対象者	2
目的	2
オペレーショナルエクセレンス	3
IaC アプローチを使用してデプロイを自動化する	3
小規模かつ可逆的な変更を頻繁に行う	4
失敗を予測する	4
すべての運用上の障害から学ぶ	5
ログ機能を使用して、不正または異常なアクティビティをモニタリングする	5
セキュリティ	7
データセキュリティの実装	7
ネットワークを保護する	8
認証と認可を実装する	8
信頼性	10
サービスクォータとデプロイパターンを含むワークロードアーキテクチャ	10
デプロイパターン	10
InfluxDB の Timestream の管理とスケーリング	11
パフォーマンス効率	13
流入データモデリングとクエリの最適化	13
書き込みの最適化	15
コスト最適化	16
ユースケースの要件とコストを理解する	16
コストに留意したリソースの選択	16
過剰な支出なしでスケールし、ビジネスニーズを満たす	17
データストレージと転送の適切なサイズ設定	17
持続可能性	19
AWS リージョン 選択	19
ユーザー動作パターンに基づくリソース消費	20
ソフトウェア開発とアーキテクチャパターンを最適化する	20
リソース	21
リファレンス	21
ブログ記事	21
ドキュメント履歴	22
.....	xxiii

Amazon Timestream for InfluxDB 用の AWS Well-Architected フレームワークの適用 InfluxDB

Balwanth Bobilli、Amazon Web Services

2025 年 1 月 ([ドキュメント履歴](#))

[Amazon Timestream](#) を使用して、Amazon Web Services (AWS) time-series-basedソリューションを構築できます。Amazon Timestream は、低レイテンシーのクエリから大規模なデータ取り込みまで、ワークロード向けにフルマネージド型の専用時系列データベースエンジンを提供します。[Amazon Timestream for InfluxDB](#) を使用すると、リアルタイムアラートやインフラストラクチャの信頼性のモニタリングなどの時系列アプリケーション AWS 用に、オープンソースの InfluxDB データベースをミリ秒の応答時間で実行できます。InfluxDB の Timestream は、最大 99.9% の可用性を提供します。

このガイドでは、Timestream for InfluxDB デプロイを計画するときに [AWS Well-Architected フレームワーク](#) の原則を適用するための規範的なガイダンスを提供します。AWS Well-Architected フレームワークは、さまざまなアプリケーションやワークロード向けに、安全で高性能、耐障害性、効率的なインフラストラクチャを構築するのに役立ちます。また、アーキテクチャを評価し、スケーラブルな設計を実装するための一貫したアプローチも提供します。

AWS Well-Architected フレームワークは、次の 6 つの柱を中心に構築されています。

- オペレーショナルエクセレンス
- セキュリティ
- 信頼性
- パフォーマンス効率
- コスト最適化
- 持続可能性

このガイドでは、AWS Well-Architected フレームワークの設計の柱について説明します。Amazon Timestream for InfluxDB をデプロイするときは、これらのベストプラクティスを使用することを検討してください AWS。

 Note

一部の AWS のサービスは、一部の AWS リージョンでは使用できません。リージョンの可用性については、AWS ドキュメントの「[サービスエンドポイントとクォータ](#)」ページを参照して、サービスのリンクを選択します。

対象者

このガイドは、時系列データのソリューションを設計および実装するデータエンジニア、ソリューションアーキテクト、データアナリストを対象としています AWS。

目的

このガイドは、お客様や組織が以下のことを行うのに役立ちます。

- サポートされているデプロイオプションから選択し、最適化された書き込みを実行し、きめ細かなアクセスを実装します。
- 耐障害性とセキュリティの向上に役立つ AWS Well-Architected 設計パターンに従ってください。
- 最適なパフォーマンスを実現するようにクエリを設計します。
- Timestream for InfluxDB インスタンスを本番環境で管理する際に運用効率を高める方法について説明します。

運用上の優秀性の柱

AWS Well-Architected フレームワークの[運用上の優秀性の柱](#)は、システムの実行とモニタリング、ビジネス価値をもたらすためのプロセスと手順の継続的な改善に焦点を当てています。運用上の優秀性の柱には、開発をサポートし、ワークロードを効果的に実行し、運用に関するインサイトを得る能力が含まれます。

人間の介入なしにほとんどの問題を検出して修復する自己修復ワークロードによって、運用上の複雑さを軽減できます。この目標を達成するために、このセクションで説明されているベストプラクティスに従ってください。Amazon Timestream for InfluxDB、InfluxDB ネイティブメトリクスエンドポイント、APIs、およびメカニズムの Amazon [Amazon CloudWatch](#) メトリクスを使用して、ワークロードが予想される動作から逸脱したときに応答します。

運用上の優秀性の柱に関するこの説明では、以下の主要分野に焦点を当てています。

- Infrastructure as code (IaC)
- 変更管理
- レジリエンシー戦略
- インシデント管理
- 監査目的のログ記録とモニタリング

IaC アプローチを使用してデプロイを自動化する

IaC を使用して Timestream for InfluxDB でのデプロイを自動化するためのベストプラクティスは次のとおりです。

- IaC を適用して、可能な限り InfluxDB の Timestream をデプロイします。一貫した環境設定を行うには、[AWS CloudFormation](#) テンプレート、[AWS Cloud Development Kit \(AWS CDK\)](#)、または [HashiCorp Terraform](#) を使用して、インスタンスに必要なすべてのリソースを作成します。
- インスタンスのサイズ変更など、InfluxDB の運用手順の Timestream を自動化します。
- タグを使用して、InfluxDB リソースの Timestream にメタデータを追加し、タグに基づいて使用状況を追跡します。詳細については、「[InfluxDB の Amazon Timestream のタグ付け](#)」を参照してください。

小規模かつ可逆的な変更を頻繁に行う

以下の推奨事項は、複雑さを最小限に抑え、ワークロードの中断の可能性を減らすための、小規模かつ可逆的な変更に焦点を当てています。

- IaC テンプレートとスクリプトを GitHub や GitLab などのソース制御サービスに保存します。ソース管理に AWS 認証情報を保存しないでください。
- IaC デプロイでは、[AWS CodeDeploy](#) や [AWS CodeBuild](#) などの継続的インテグレーションおよび継続的デリバリー (CI/CD) サービスを使用する必要があります。これらのサービスは、本番環境の InfluxDB インスタンスに影響を与える前に、一時的な InfluxDB インスタンスを含む非本番環境でコードをコンパイル、テスト、デプロイします。
- 本番環境にデプロイする前に、インフラストラクチャとアプリケーションのクエリをより低い環境でテストします。これにより、中断の可能性が最小限に抑えられ、ワークロードやスケールに合わせて適切に動作するようになります。

失敗を予測する

自己修復インフラストラクチャは、障害を予測し、介入なしで問題解決を試みることで、運用上の優秀性を実証します。以下の推奨事項は、Timestream for InfluxDB でその成熟度を達成するのに役立ちます。

- メトリクスを使用して、メモリ、CPU、ストレージの使用状況をモニタリングします。使用パターンが変化したとき、またはデプロイの容量に近づいたときに通知するように CloudWatch を設定できます。このようにして、システムのパフォーマンスと可用性を維持できます。
- リソース制限に近づいたら、DB インスタンスをスケールアップします。アプリケーションからのリクエストの予期しない増加に対応するために、ストレージとメモリにいくらかのバッファがあることが必要です。
- データベースのワークロードでプロビジョニングした I/O がより多く必要になると、フェイルオーバーやデータベース障害後の復旧が遅くなります。DB インスタンスの I/O 容量を増やすには、I/O 容量が大きい別の DB インスタンスに移行します。
- クライアントアプリケーションが DB インスタンスの DNS データをキャッシュしている場合は、time-to-live (TTL) 値を 30 秒未満に設定します。DB インスタンスの基になる IP アドレスは、フェイルオーバー後に変更されている可能性があります。DNS データを長期間キャッシュすると、接続が失敗する可能性があります。アプリケーションが、使用されなくなった IP アドレスに接続しようとする可能性があります。

- アプリケーションが完全に AWS リージョン 停止した場合、ディザスタリカバリ (DR) プランの一部としてレプリケーションを設定するか、別のリージョンに書き込むことを検討してください。レプリケーションをセットアップする際の制限を理解します。レプリケーションの詳細については、[InfluxDB ドキュメント](#)を参照してください。

すべての運用上の障害から学ぶ

自己修復インフラストラクチャは、まれな問題が発生したり、対応が意図したほど効果的でない場合に反復して開発する長期的な労力です。自己修復インフラストラクチャの達成に集中するには、次のプラクティスを採用します。

- すべての障害から学ぶことで改善を推進します。
- チームと組織で教訓を共有します。組織内の複数のチームが Timestream for InfluxDB を使用している場合は、共通のチャットルームまたはユーザーグループを作成して、教訓とベストプラクティスを共有します。

ログ機能を使用して、不正または異常なアクティビティをモニタリングする

異常なパフォーマンスとアクティビティのパターンを観察するには、次のプラクティスを検討してください。

- [ログ配信](#)を有効にして、InfluxDB ログを [Amazon Simple Storage Service \(Amazon S3\)](#) に保存します。InfluxDB は、以下を確認するのに役立つレコード情報をログに記録します。
 - [データプレーン API イベント](#)
 - 応答時間
 - 圧縮の詳細
 - システムで発生した重大なエラーまたは警告

ログに不正アクセスや異常がないか確認します。全体として、ログ記録はトラブルシューティングの診断情報を提供します。

- InfluxDB の Timestream は、を使用したコントロールプレーンアクションのログ記録をサポートしています AWS CloudTrail。詳細については、「[Logging Timestream for InfluxDB API calls with AWS CloudTrail](#)」を参照してください。

- CloudWatch の Timestream/InfluxDB > <Namespace> から
CPUUtilizationMemoryUtilization、DiskUtilizationメトリクスをモニタリングできます。

詳細については、[「Timestream for InfluxDB ドキュメント」](#)を参照してください。

セキュリティの柱

[セキュリティの柱](#)は、Timestream for InfluxDB を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。

セキュリティの柱には、以下の主要な重点分野が含まれています。

- データセキュリティ
- ネットワークセキュリティ
- 認証と認可

以下のセクションでは、セキュリティおよびコンプライアンスの目的を達成するように Timestream for InfluxDB を設定する方法を示します。また、Timestream for InfluxDB リソースのモニタリングと保護 AWS のサービス に役立つ他の の使用方法についても説明します。

データセキュリティの実装

データ漏洩やデータ侵害は顧客を危険にさらし、会社に大きな悪影響を与える可能性があります。責任 AWS [共有モデル](#)は、Timestream for InfluxDB でのデータ保護に適用されます。以下のプラクティスは、不注意や悪意のある漏洩から顧客データを保護するのに役立ちます。

- インスタンス名、タグ、パラメータグループ、AWS Identity and Access Management (IAM) ロール、およびその他のメタデータに機密情報を含めないでください。そのデータは、請求ログまたは診断ログに表示される場合があります。
- 暗号化を使用して、クラウドにデプロイされているアプリケーションのデータ保護を強化します。暗号化されたインスタンスは、基盤となるストレージへの不正アクセスからデータを保護するのに役立つため、データ保護の追加レイヤーを提供します。InfluxDB 暗号化の Timestream はデフォルトで有効になっています。
- 可能な限り APIs のパラメータ化を使用します。

データ保護と暗号化の詳細については、[Timestream for InfluxDB ドキュメント](#)を参照してください。

ネットワークを保護する

InfluxDB インスタンスの Timestream は、の仮想プライベートクラウド (VPC) でのみ作成できます AWS。インスタンスをさらに保護するには、以下を実行します。

- 次のいずれかを実行して、インスタンスのエンドポイント へのパブリックアクセスを制限します。
 - InfluxDB データベースの作成ページでパブリックにアクセスできない (デフォルトで選択) を選択します。
 - [PubliclyAccessible](#) プロパティを `()false` で AWS CloudFormation に設定します `AWS::Timestream::InfluxDBInstance`。

パブリックにアクセスできないまたは `PubliclyAccessible` に設定すると `false`、インスタンスのエンドポイントは VPC 内でのみアクセスできます。エンドポイントは通常、Timestream for InfluxDB インスタンスと同じ VPC で実行されている [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) インスタンスからアクセスされます。InfluxDB

- セキュリティグループを使用して、VPC 内の Timestream for InfluxDB へのネットワークアクセスをさらに保護します。
- SSL/TLS を使用して AWS リソースと通信します。InfluxDB の Timestream には TLS 1.2 が必要です。TLS 1.3 をお勧めします。
- 「[InfluxDB インスタンスの Timestream を作成して接続する](#)」の手順に従って、**インスタンスに安全に接続**します。
- [インターフェイス](#) VPC エンドポイントを作成して、VPC と Amazon Timestream for InfluxDB コントロールプレーン API エンドポイント間のプライベート接続を確立します。

セキュリティの詳細については、[「Timestream for InfluxDB ドキュメント」](#)を参照してください。

認証と認可を実装する

IAM 認証情報を使用して、Timestream for InfluxDB インスタンスの管理 アクションを制御します。IAM 認証情報を使用して Timestream for InfluxDB に接続する場合、IAM ロールには、Timestream for InfluxDB 管理オペレーションの実行に必要なアクセス許可を付与する IAM ポリシーが必要です。最小特権の原則に従い、タスクを完了するために必要なアクセス許可のみを付与してください。詳細については、「[Identity and Access Management for Amazon Timestream for InfluxDB](#)」を参照してください。

IAM ポリシーで [Amazon Timestream for InfluxDB アクション](#) を使用して、Timestream for InfluxDB インスタンスを管理できるユーザーを制御できます。

Amazon Timestream for InfluxDB インスタンスに保存されているデータのきめ細かなアクセスコントロールを提供するには、ユーザーに [InfluxDB API トークンを付与します](#)。

他の とやり取りするために AWS のサービス、Amazon Timestream for InfluxDB は IAM サービスにリンクされたロールを使用します。サービスにリンクされたロールは、Timestream for InfluxDB に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、InfluxDB の Timestream によって事前定義されており、サービスが AWS のサービス ユーザーに代わって他の を呼び出すために必要なすべてのアクセス許可が含まれています。詳細については、「[Using Service-Linked Roles for Amazon Timestream for InfluxDB](#)」を参照してください。

信頼性の柱

[信頼性の柱](#)には、ワークロードが意図した機能を正しく一貫して実行する能力が含まれます。これには、ライフサイクル全体を通じてワークロードを運用およびテストする機能が含まれます。

信頼性の高いワークロードの設定は、ソフトウェアとインフラストラクチャの両方の設計に関する事前の決定から始まります。アーキテクチャの選択は、Well-Architected の柱のすべてにおいて、ワークロードの動作に影響を与えます。信頼性を実現するには、特定のパターンに従う必要があります。

信頼性の柱は、次の主要分野に焦点を当てています。

- サービスクォータとデプロイパターンを含むワークロードアーキテクチャ
- InfluxDB インスタンスの管理とスケーリング

サービスクォータとデプロイパターンを含むワークロードアーキテクチャ

各 AWS アカウント には、各 で提供されるリソースのクォータがあります AWS リージョン。たとえば、各リージョンには、インスタンスサイズに関係なく、[InfluxDB インスタンスの Timestream のクォータ](#)があります。リージョン内のインスタンスの最大数に達すると、インスタンスを作成するための追加の呼び出しは例外で失敗します。InfluxDB インスタンスストレージボリュームの Timestream は、サポートされているすべての で最大サイズ 16 テビバイト (TiBs) まで拡張できます AWS リージョン。

デプロイパターン

InfluxDB インスタンスの Timestream の[高可用性](#)とフェイルオーバーのサポートのために、単一のスタンバイ DB インスタンスでマルチ AZ 配置を使用できます。このタイプのデプロイは、マルチ AZ DB インスタンスのデプロイと呼ばれます。Amazon Timestream for InfluxDB は、Amazon フェイルオーバーテクノロジーを使用します。マルチ AZ DB インスタンスのデプロイでは、Amazon Timestream は、異なるアベイラビリティゾーンで同期スタンバイレプリカを自動的にプロビジョニングおよび維持します。データの冗長性を提供するために、プライマリ DB インスタンスはアベイラビリティゾーン間でスタンバイレプリカに同期的にレプリケートされます。

高可用性を備えた DB インスタンスを実行すると、DB インスタンスの障害時やアベイラビリティゾーンの中断時に可用性が提供される可能性があります。DB インスタンスの予期しない停止がイン

フラストラクチャの欠陥に起因する場合、Amazon Timestream for InfluxDB は自動的にスタンバイレプリカに切り替わります。フェイルオーバーが完了するまでにかかる時間は、プライマリ DB インスタンスが使用できなくなったときのデータベースアクティビティおよびその他の条件によって異なります。

フェイルオーバー時間は通常 60～120 秒です。ただし、カーディナリティの高いデータを含む大規模なトランザクションや、ウォームアップ前の要件を含む長い復旧プロセスでは、フェイルオーバー時間が長くなる可能性があります。フェイルオーバーが完了したら、Timestream コンソールに新しいアベイラビリティゾーンが反映されるまでに追加の時間が必要になる場合があります。

完全な AWS リージョン 停止中にアプリケーションを使用できるようにする必要がある場合は、ディザスタリカバリ (DR) プランの一部としてレプリケーションを設定するか、別のリージョンに書き込むことを検討してください。ただし、レプリケーションを設定する前に、必ず制限事項を理解してください。詳細については、[InfluxDB ドキュメント](#)を参照してください。

Amazon Timestream for InfluxDB は、可用性と耐久性をサポートするため、内部バックアップを定期的に取得してこれを 24 時間保持します。削除中はスナップショットが作成され、復元をサポートするために 30 日間保持されます。これらにアクセスまたは使用するには、[ケースを作成します](#) [AWS サポート](#)。

InfluxDB の Timestream の管理とスケーリング

InfluxDB の Timestream は、オープンソースの InfluxDB データベースでメモリを大量に消費するワークロードを実行するのに最適なインスタンスクラスをサポートしています。異なる [db.influx インスタンスクラス](#)には、vCPUsメモリ、ストレージ、ネットワーク帯域幅に制限があります。アプリケーションの書き込みおよびクエリのレイテンシー要件に合ったインスタンスクラスを選択するには、テスト中に Amazon CloudWatch CPUUtilization、MemoryUtilization、および DiskUtilizationメトリクスを確認します。ワークロードの要件に基づいて[インスタンスをスケールアップまたはスケールダウン](#)できます。InfluxDB の Timestream は、さまざまなタイプのワークロードに必要な最適な IOPS とスループットで事前設定された複数のストレージ層を提供します。要件に基づいて、ワークロードに最適なものを選択します。

スケーリングのニーズが予測可能な時間に変化する場合は、[AWS Lambda 関数](#)またはカスタムスケジューラを使用して API または SDK を実行して、バッファ時間を指定してスケールアップまたはスケールダウンできます。

パラメータグループのパラメータを使用して、Timestream for InfluxDB で InfluxDB InfluxDB 設定を管理します。パラメータグループは、1 つ以上の DB インスタンスに適用される InfluxDB 設定オブ

シヨンのコンテナとして機能します。パラメータグループのパラメータを変更するときは、静的パラメータと動的パラメータの違いと、それらがどのようにいつ適用されるかを理解します。現在適用されている設定を確認するには、[GetDbParameterGroup](#) API アクションを使用します。

パフォーマンス効率の柱

AWS Well-Architected フレームワークの[パフォーマンス効率の柱](#)は、データの取り込みまたはクエリ中にパフォーマンスを最適化する方法に焦点を当てています。パフォーマンスの最適化は、以下の段階的かつ継続的なプロセスです。

- ビジネス要件の確認
- ワークロードのパフォーマンスの測定
- パフォーマンスの低いコンポーネントを特定する
- ビジネスニーズに合わせてコンポーネントを調整する

パフォーマンス効率の柱は、高パフォーマンスのデータモデルを選択するのに役立つガイドラインを提供します。パフォーマンス効率の柱には、クエリと書き込みの最適化のベストプラクティスが含まれます。

パフォーマンス効率の柱は、次の主要分野に焦点を当てています。

- 流入データモデリングとクエリの最適化
- 書き込みの最適化

流入データモデリングとクエリの最適化

InfluxDB の時系列データのパフォーマンスとクエリ機能を最適化するには、効果的なスキーマの設計が不可欠です。まず、適切なタグとフィールドを選択します。InfluxDB はタグをインデックス化するため、クエリエンジンは測定値内のすべてのレコードをスキャンしてタグ値を見つける必要はありません。つまり、タグのクエリは、フィールドのクエリよりも効率的です。データを圧縮して保存するために、ストレージエンジンはフィールド値を系列キーでグループ化し、それらのフィールド値を時間順に並べます。シリーズキーは、測定、タグキーと値、およびフィールドキーによって定義されます。データ設計の詳細については、[InfluxDB ドキュメント](#)を参照してください。

ストレージエンジンは、時間構造化マージツリー (TSM) データ形式を使用します。TSM データ形式の詳細については、[InfluxDB ドキュメント](#)を参照してください。

DevOps ユースケースの一部としてデータを収集するとします

(timestamp、host_id、region、cpu、memorynetwork_in_bytesnetwork_out_bytes、disk_i

レコードのタイムスタンプを含むタグは、レコードのユーザー、内容、日時、場所を識別するのに役立つコンテキストを提供します。タグは、データを整理および分類し、クエリの一部としてデータをフィルタリングするために使用されます。

host_id および region タグは、DevOps ユースケースの整理と分類に最適なタグです。これらの列は、特定のホストのデータをフィルタリングしたり、リージョン列に基づいて分析を実行したりするのに役立ちます。

メジャーは、データに対して数学的計算 (合計、平均、変化率の差の計算など) と定量的分析を実行するための基盤を提供します。したがって、cpu、memorynetwork_in_bytes、および network_out_bytes、時間の経過とともに変化する DevOps に関連する重要なメトリクスを disk_io キャプチャします。これらのメトリクスを使用して、異なるホスト間で CPU とメモリを計算するなど、さまざまな分析を実行できます。これらのメトリクス値を使用して、本番稼働停止の回避やインフラストラクチャ計画の実行に役立つデータ駆動型意思決定を行うことができます。

Cardinality は一意のタグ値の組み合わせです。カーディナリティをできるだけ低く維持することを目指します。アプリケーションでデータポイントごとに一意の識別子が必要な場合は、タグ値の代わりにフィールド値を使用します。これにより、クエリのレイテンシーが大幅に向上します。スキーマを適切に設計すると、系列のカーディナリティが高くなり、クエリの実行が向上します。データの読み取りと書き込みが遅くなることに気付いた場合、またはカーディナリティがパフォーマンスにどのように影響するかを確認するには、[「Timestream for InfluxDB ドキュメント」](#)を参照してください。

アプリケーションが JSON オブジェクトを出力する場合は、それらを個々の列 (タグまたはフィールド) に変換し、列を InfluxDB にロードします。InfluxDB は時系列データ用に設計されているため、個々の列でデータを整理することが、サービスの機能を最大限に活用するためのベストプラクティスです。

1 つの InfluxDB v2.7 OSS インスタンスは、すべての組織間でアクティブに書き込みまたはクエリされている約 20 個の InfluxDB バケットをサポートします。20 を超えるバケットがパフォーマンスに悪影響を及ぼす可能性があります。InfluxDB の設定オプションには制限があり、ユースケースに基づいて設定できるオプションもあります。テストフェーズでアプリケーションワークロードに基づいて設定を検証します。データ保持はバケットレベルで設定されるため、異なるデータ保持要件を持つデータは異なるバケットに保存する必要があります。設定オプションの詳細については、[「Timestream for InfluxDB ドキュメント」](#)を参照してください。

データは、タグキー、フィールドキー、測定値ではなく、タグ値またはフィールド値に保存します。タグ値とフィールド値にデータを保存するようにスキーマを設計すると、クエリの書き込みが容易になり、効率が向上します。データモデリングのベストプラクティスの詳細については、[「パフォーマンスのための設計」](#)を参照してください。

[InfluxDB タスク](#)を使用して、データを事前集計し、データをさまざまな測定値またはバケットにロードし、そこからダッシュボードとビジュアライゼーションのデータを生成します。

InfluxDB OSS は、Prometheus プレーンテキスト表示形式でフォーマットされたパフォーマンス、リソース、使用状況メトリクスを返す `/metrics` [エンドポイント](#) を公開します。InfluxDB テンプレートを使用して [モニタリングとアラート](#) を設定し、高いクエリレイテンシー、書き込みスループットの低下、リソース使用量の急増などの問題を事前に検出します。

InfluxDB の Timestream は、Influx IO 込みストレージを提供します。適切な IOPS サイズを選択すると、クエリの実行が大幅に高速化されます。これは、大量のデータをスキャンしたり、幅広いリクエストを処理する必要があるクエリに特に役立ちます。状況によっては、必要なパフォーマンスの向上を実現するために、インスタンスのスケールアップと IOPS の強化の組み合わせが必要になる場合があります。

dev 環境と prod 環境 (インスタンスクラス、ストレージタイプ、設定) を一致させることをお勧めします。本番環境に移行する前に、リリースごとに下位環境の変更をテストします。Influx IO 搭載ストレージボリュームでは、Timestream for InfluxDB は、さまざまなタイプのワークロードに必要な最適な IOPS (3,000、12,000、16,000) とスループットで事前設定された 3 つのストレージ層を提供します。ほとんどのユースケースでは、必要な IOPS は 3,000 未満です。パフォーマンステストで高い IOPS の必要性が示されている場合にのみ、12,000 または 16,000 を選択します。詳細については、「Timestream for InfluxDB ドキュメント」の [「セットアップ」セクション](#) を参照してください。

書き込みの最適化

InfluxDB への書き込みを最適化するには、ネットワークオーバーヘッドを最小限に抑えるために、リクエストごとに 5,000 行のラインプロトコルのバッチでデータを書き込むことをお勧めします。パフォーマンスを向上させるには、データポイントを書き込む前に、キーで辞書順にタグをソートします。ナノ秒ではなく、タイムスタンプに可能な限り粗い時間精度を使用すると、パフォーマンスも向上します。gzip 圧縮を有効にすることは、書き込みを高速化し、ネットワーク帯域幅を減らすもう 1 つの方法です。telegraf.conf ファイルの influxdb_v2 出力プラグイン設定で、content_encoding オプションを に設定します gzip。これらの最適化を実装すると、InfluxDB にデータを書き込むパフォーマンスと効率が大幅に向上します。InfluxDB 書き込みのベストプラクティスの詳細については、「[InfluxDB への書き込みの最適化](#)」を参照してください。

InfluxDB の書き込みパフォーマンスは、多くの場合、使用可能な IOPS と密接に関連しています。データを書き込む場合、InfluxDB は大量の I/O オペレーションを実行してデータを保存する必要があります。IOPS を増やすと、InfluxDB は 1 秒あたりの書き込み数を増やすことができます。

コスト最適化の柱

AWS Well-Architected フレームワークの[コスト最適化の柱](#)は、不要なコストを回避し、コスト最適化の方法でアーキテクチャを構築することに重点を置いています。以下の推奨事項は、Amazon Timestream for InfluxDB のコスト最適化設計原則とアーキテクチャのベストプラクティスを満たすのに役立ちます。

コスト最適化の柱は、次の主要分野に焦点を当てています。

- ユースケースの要件とコストを理解する
- コストに留意したリソースの選択
- 過剰な支出なしでスケールし、ビジネスニーズを満たす
- データストレージと転送の適切なサイズ設定

ユースケースの要件とコストを理解する

以下のユースケースでは、InfluxDB に Timestream を使用しないことをお勧めします。

- データモデルにリレーショナルデータがある場合、Timestream for InfluxDB は適切なソリューションではありません。
- クエリで時間フィルターを使用できない場合、Influx はすべてのシリーズをスキャンします。これは非効率です。

コストに留意したリソースの選択

[InfluxDB インスタンス](#)のコストは、インスタンスが実行される時間の時間単位のレートに基づきます。インスタンスは、データベースの実行にかかる総コストの平均 85% を占めるため AWS、適切なサイジングはコストに大きな影響を与える可能性があります。インスタンスのサイズを適正化する最善の方法は、アプリケーションのパフォーマンスをテストすることです。

- CPUUtilization とは常にMemoryUtilization高いか低い。
- 価格とパフォーマンスのバランスはどれくらいですか？

インスタンスのコストは直線的にスケールされます。db.influx.2xlarge インスタンスの時間単位のコストはdb.influx.xlargeインスタンスの 2 倍ですが、リソース割り当ても 2 倍で

す。db.influx.16xlarge インスタンスは、db.influx.xlarge インスタンスの時間単位のコストの 16 倍です。

特定の時間枠 (秒、分、時間、または日) におけるワークロードの書き込みと読み取りの数を推定します。InfluxDB インスタンスの Timestream は、インスタンスタイプに基づいて 1 秒あたり 50,000 ~ 500,000 回の書き込みと 1 秒あたり 10 ~ 100 回のクエリ (QPS) をサポートします。たとえば、db.influx.2xlarge は通常、1 秒あたり最大 150,000 回の書き込みと約 25 QPS をサポートします。効率的なデータモデルと効率的なクエリにより、そのパフォーマンスを超える可能性があります。要件が時刻、週、または月によって異なる場合は、以下を実行してスケールアップとスケールダウンをスケジュールできます。

- [AWS Lambda 関数を作成してスケジュールします](#)。
- カスタムスケジューラを使用して API または SDK を実行し、バッファ時間で [スケールアップおよびスケールダウン](#) します。

過剰な支出なしでスケールし、ビジネスニーズを満たす

Timestream for InfluxDB のエントリレベルの実験では、db.influx.medium とを使用してきます db.influx.large。これらのインスタンスは、大規模なインスタンスに投資する前に、Timestream for InfluxDB の使用経験を積むのに十分な大きさです。

db.influx.medium および db.influx.large インスタンスは、低コストの開発環境に適しています。ただし、RAM (8 GiB と 16 GiB) は小さく vCPUs (1 vCPU と 2 vCPUs) は少なく、ネットワークパフォーマンスは最大 10 GB です。すべてのワークロードがこれらのインスタンスクラスに適しているわけではありません。CPU Utilization と Memory Utilization をモニタリングし、必要に応じてスケールアップまたはスケールダウンします。通常、メモリと vCPU の比率は一定です。db.influx インスタンスクラスは、Amazon EC2 r7g インスタンスクラスとメモリ memory-to-vCPU の比率が似ています。本番稼働を開始する前に end-to-end のパフォーマンステストまたは負荷テストを実行することを強くお勧めします。

効率的なデータモデリング、バッチ書き込み、最適化されたクエリでは、メモリとコンピューティングの使用量が少なく済みます。必要なリソースが少ない場合は、使用するインスタンスが小さくなる可能性があります。

データストレージと転送の適切なサイズ設定

データを保存するには、次のベストプラクティスを使用します。

- InfluxDB の Timestream には時系列データのみを保存します。
- InfluxDB バケットに適切な保持を設定し、保持期間より古いデータが削除され、シャードが定期的に自動的に圧縮されるようにします。詳細については、[InfluxDB ドキュメント](#)を参照してください。
- 将来の書き込み用にディスク使用量を最適化します。
- ワークロードに必要な InfluxDB バケットを削除します。InfluxDB は削除をサポートしています。ユースケースに合った場合は、スケジュールされたクリーンアップを実行できます。

データ転送では、クロスリージョンネットワークオーバーヘッドを避けるため、Timestream for InfluxDB データベースインスタンス AWS リージョン と同じ にアプリケーションをデプロイすることをお勧めします。データ転送料金が発生する場合があります。データ転送の詳細については、「[料金表](#)」ページを参照してください。

持続可能性の柱

[持続可能性の柱](#)は、クラウドワークロードの実行による環境への影響を最小限に抑えることに焦点を当てています。持続可能性の柱には、以下の主要な重点分野が含まれています。

- 影響を理解する
- 持続可能性の目標
- リソースを最小限に抑えるための使用の最大化
- より効率的な新しいハードウェアおよびソフトウェア製品の予測および採用
- マネージドサービスの使用
- ダウンストリームへの影響の軽減

このガイドでは、影響の理解に焦点を当てています。その他の持続可能性設計原則の詳細については、[AWS「Well-Architected フレームワーク」](#)を参照してください。

ユーザーによる選択と要件は、環境に影響を及ぼします。ワークロードの持続可能性を高めるには、以下を実行します。

- 炭素強度が低い AWS リージョン ものを選択します。
- 稼働時間と耐久性を最大化するのではなく、実際のワークロードのニーズを反映するようにリソースのサイズを設定します。
- データモデルを最適化し、コンピューティングリソースの使用を最大化します。

次のセクションでは、ワークロード設計と継続的な運用における環境への影響を軽減するために採用できるプラクティスについて説明します。

AWS リージョン 選択

一部の AWS リージョン は Amazon 再生可能エネルギープロジェクトの近くにあるか、グリッドの公開された炭素強度が他のグリッドよりも低い場所にあります。[持続可能性の目標](#)とワークロードの要件に基づいてリージョンを評価します。次に、有効なリージョンのリストを、[Timestream for InfluxDB が利用可能なリージョン](#)と相互参照します。

ユーザー動作パターンに基づくリソース消費

ユーザーのトラフィックと動作に合わせて使用量を適切なサイズにすることで、AWS が環境に与える影響を最小限に抑えることができます。ソリューションを設計するときは、次のベストプラクティスを考慮してください。

- CPUUtilization や などの Amazon CloudWatch メトリクスをモニタリング MemoryUtilization して、需要が最も高いタイミングと最も低いタイミングを判断します。その間は、インスタンスリソースのサイズが正しいことを確認してください。
- サービスレベル契約を、ビジネス継続性の目標だけでなく持続可能性の目標に合わせることを検討してください。マルチリージョンディザスタリカバリ、高可用性、長期バックアップ保持などの要件を緩和することで、これらの目標を達成するために必要なリソースの量を減らすことができます。非本番環境と非ミッションクリティカルなワークロードは、要件を削減する機会を提供します。

ソフトウェア開発とアーキテクチャパターンを最適化する

無駄を防ぐには、データモデルとクエリを最適化します。コンピューティングリソースを共有して、Timestream for InfluxDB インスタンスで利用できるすべてのリソースを使用します。以下のベストプラクティスを実装することをお勧めします。

- Timestream for InfluxDB スタックを共有して、可能な限りリソースの使用を改善するよう、デベロッパーチームを奨励します。
- リソースの使用を最大化し、アイドル時間を最小限に抑えるパターンを実装します。パターンの例には、並列スレッドを使用してデータをロードし、レコードをまとめてより大きなトランザクションにバッチ処理することが含まれます。
- クエリと InfluxDB データモデルを最適化して、結果の計算に必要なリソースを最小限に抑えます。
- [InfluxDB タスク](#) を使用してデータを事前に集約し、視覚化またはダッシュボード作成のために異なるユーザーによる同じ raw データのスキャンを減らします。
- Timestream for InfluxDB 環境を最新の状態に保ちます。最新バージョンの Timestream for InfluxDB は、Graviton などの最新の EC2 インスタンスをより効率的にサポートします。最新の DB バージョンには、クエリの最適化の改善と、クエリの計算に必要なリソースの量を減らすバグ修正も含まれています。

リソース

リファレンス

- [AWS Well-Architected](#)
- [AWS Well-Architected フレームワークのドキュメント](#)
- [Amazon Timestream for InfluxDB ドキュメント](#)
- [InfluxDB OSS v2 ドキュメント](#)

ブログ記事

- [AWS InfluxDB 移行スクリプトを使用して、InfluxDB OSS 2.x データを InfluxDB 用 Amazon Timestream に移行する](#)
- [Amazon Timestream を使用してオープンソースの InfluxDB データベースを実行および管理する](#)

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、[RSS フィード](#) をサブスクライブできます。

変更	説明	日付
初版発行	—	2025 年 1 月 28 日

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。