でのマルチテナント SaaS アプリケーション用のマネージド PostgreSQL の実装 AWS

AWS 規範ガイダンス

AWS 規範ガイダンス: でのマルチテナント SaaS アプリケーション用のマネージド PostgreSQL の実装 AWS

Table of Contents

でのマルチテナント SaaS アプリケーション用のマネージド PostgreSQL の実装 AWS

Tabby Ward と Thomas Davis、Amazon Web Services (AWS)

2024 年 4 月 (ドキュメント履歴)

運用データを保存するデータベースを選択するときは、データの構造化方法、回答するクエリ、回答を提供する速度、データプラットフォーム自体の耐障害性を考慮することが重要です。これらの一般的な考慮事項に加えて、Software as a Service (SaaS) は、パフォーマンスの分離、テナントセキュリティ、マルチテナント SaaS アプリケーションのデータに典型的な固有の特性と設計パターンなど、運用データに影響します。このガイドでは、これらの要因が、マルチテナント SaaS アプリケーションのプライマリ運用データストアとして Amazon Web Services (AWS) の PostgreSQL データベースを使用する場合にどのように適用されるかについて説明します。具体的には、このガイドでは、Amazon Aurora PostgreSQL -Compatible Edition と PostgreSQL Amazon Relational Database Service RDS) for PostgreSQL の 2 つの AWS マネージド PostgreSQL オプションに焦点を当てています。 PostgreSQL

ターゲットを絞ったビジネス成果

このガイダンスでは、Aurora PostgreSQL 互換および Amazon RDS for PostgreSQL を使用したマルチテナント SaaS アプリケーションのベストプラクティスの詳細な分析を提供します。このガイドに記載されている設計パターンと概念を使用して、マルチテナント SaaS アプリケーション用の Aurora PostgreSQL 互換または Amazon RDS for PostgreSQL の実装を通知および標準化することをお勧めします。

この規範的なガイダンスは、以下のビジネス成果を達成するのに役立ちます。

- ユースケースに最適な AWS マネージド PostgreSQL オプションの選択 — このガイダンスでは、SaaS アプリケーションでデータベースを使用するためのリレーショナルオプションと非リレーショナルオプションを比較します。また、Aurora PostgreSQL -Compatible および Amazon RDS for PostgreSQL に最適なユースケースについても説明します。この情報は、SaaS アプリケーションに最適なオプションを選択するのに役立ちます。
- SaaS パーティショニングモデルの導入による SaaS ベストプラクティスの実施 – このガイドでは、PostgreSQL データベース管理システム (DBMS) に適用可能な 3 つの広範な SaaS パーティショニングモデル、プール、ブリッジ、サイロモデル、およびそれらのバリエーションについて説

明し、比較します。これらのアプローチは、SaaS のベストプラクティスをキャプチャし、SaaS アプリケーションを設計する際の柔軟性を提供します。SaaS パーティショニングモデルの適用は、ベストプラクティスを維持する上で重要な部分です。

- プール SaaS パーティショニングモデルでの RLS の効果的な使用 – 行レベルのセキュリティ (RLS) は、ユーザーまたはコンテキスト変数に基づいて表示できる行を制限することで、単一の PostgreSQL テーブル内でのテナントデータの分離の適用をサポートします。プールパーティショニングモデルを使用する場合、クロステナントアクセスを防ぐために RLS が必要です。

SaaS アプリケーションのデータベースの選択

多くのマルチテナント SaaS アプリケーションでは、運用データベースの選択をリレーショナルデータベースと非リレーショナルデータベースのどちらにするか、またはこれら 2 つの組み合わせに分割できます。意思決定を行うには、以下の高レベルのアプリケーションデータ要件と特性を考慮してください。

- アプリケーションのデータモデル
- データのアクセスパターン
- データベースのレイテンシー要件
- データ整合性とトランザクション整合性の要件 (アトミック性、整合性、分離性、耐久性、ACID)
- リージョン間の可用性と復旧の要件

次の表は、アプリケーションデータの要件と特性を一覧表示し、 AWS データベース提供のコンテキストでそれらについて説明します。Aurora PostgreSQL -Compatible および Amazon RDS for PostgreSQL (リレーショナル) および Amazon DynamoDB (非リレーショナル) 。このマトリックスは、リレーショナル運用データベースと非リレーショナル運用データベースのどちらを提供するかを決定しようとしているときに参照できます。

データベース	SaaS アプリケーションデータの要件と特性				
	データモデル	アクセスパターン	レイテンシー要件	データおよびトランザクションの整合性	リージョン間の可用性とリカバリ
リレーショナル (Aurora PostgreSQL - 互換および Amazon RDS for PostgreSQL)	リレーショナルまたは高度に正規化されている。	事前に徹底的に計画する必要はありません。	はレイテンシー耐性を高めます。Aurora ではデフォルトで、リードレプリカ、キャッシュ、および同様の機能を	デフォルトでは、高いデータとトランザクションの整合性が維持されます。	Amazon RDS では、クロスリージョンスケーリングとフェイルオーバー用のリードレプリカを作成できます。Aurora

			実装することで、レイテンシーを低く抑えることができます。		は主にこのプロセスを自動化します。複数のにわたるアクティブ/アクティブ設定では AWS リージョン、書き込み転送を Aurora グローバルデータベース と組み合わせて使用できます。
非リレーショナル (Amazon DynamoDB)	通常、非正規化されています。これらのデータベースは、関係 、大きな項目 many-to-many 、時系列データ をモデル化するためのパターンを活用します。	データモデルを生成する前に、データのすべてのアクセスパターン (クエリ) を完全に理解する必要があります。	Amazon DynamoDB Accelerator (DAX) などのオプションにより、レイテンシーが非常に低く、パフォーマンスをさらに向上させることができます。	パフォーマンスを犠牲にしたオプションのトランザクション整合性。データ整合性に関する懸念はアプリケーションに移行されます。	グローバルテーブルによるクロスリージョンリカバリとアクティブ/アクティブ設定が容易。(ACID コンプライアンスは 1 つの AWS リージョンでのみ実現可能です。)

一部のマルチテナント SaaS アプリケーションには、前の表に含まれていないデータベースによってより適切に機能する一意のデータモデルや特殊な状況がある場合があります。例えば、時系列データセット、高度に接続されたデータセット、集中型トランザクション台帳の維持には、異なるタイプのデータベースの使用が必要になる場合があります。すべての可能性を分析することは、このガイドの

範囲外です。 AWS データベース提供の包括的なリストと、さまざまなユースケースを大まかに満たす方法については、Amazon Web Services の概要ホワイトペーパーの「データベース」セクションを参照してください。

このガイドの残りの部分では、PostgreSQL をサポートする AWS リレーショナルデータベースサービス、Amazon RDS および Aurora PostgreSQL 互換に焦点を当てています。DynamoDB では、SaaS アプリケーションの最適化に異なるアプローチが必要ですが、これはこのガイドの範囲外です。DynamoDB の詳細については、 AWS ブログ記事「Partitioning Pooled Multi-Tenant SaaS Data with Amazon DynamoDB」を参照してください。

Amazon RDS と Aurora の選択

ほとんどの場合、Amazon RDS for PostgreSQL よりも Aurora PostgreSQL -Compatible を使用することをお勧めします。次の表は、これら 2 つのオプションのどちらを選択するかを決める際に考慮すべき要素を示しています。

DBMS コンポーネント	Amazon RDS for PostgreSQL	Aurora PostgreSQL 互換
スケーラビリティ	レプリケーションラグは分、最大 5 リードレプリカ	1 分未満のレプリケーションラグ (通常はグローバルデータベースでは 1 秒未満) 、最大 15 個のリードレプリカ
クラッシュリカバリ	チェックポイントを 5 分間隔で (デフォルトで) 、データベースのパフォーマンスが低下することがある	高速リカバリのための並列スレッドによる非同期リカバリ
フェイルオーバー	クラッシュ復旧時間に加えて 60 ~ 120 秒	通常、約 30 秒 (クラッシュリカバリを含む)
[Storage (ストレージ)]	最大 IOPS は 256,000 です	Aurora インスタンスのサイズと容量によってのみ制約される IOPS
高可用性とディザスタリカバリ	スタンバイインスタンスを持つ 2 つのアベイラビリティーゾーン、リードレプリカまた	デフォルトでは 3 つのアベイラビリティーゾーン、Aurora グローバルデータベースによ

DBMS コンポーネント	Amazon RDS for PostgreSQL	Aurora PostgreSQL 互換
	はコピーされたバックアップへのクロスリージョンフェイルオーバー	るクロスリージョンフェイルオーバー、アクティブ/アクティブ設定 AWS リージョンの 間での書き込み転送
バックアップ	バックアップウィンドウ中、はパフォーマンスに影響を与える可能性があります	自動増分バックアップ、パフォーマンスへの影響なし
データベースインスタンスクラス	Amazon RDS インスタンスクラスのリストを参照	Aurora インスタンスクラスのリストを参照

前の表で説明したすべてのカテゴリでは、Aurora PostgreSQL -Compatible が通常、より良いオプションです。ただし、Amazon RDS for PostgreSQL は、Aurora のより堅牢な機能セットを犠牲にして、よりコスト効率の高いオプションを提供する可能性のあるインスタンスクラスの選択が多いため、小規模から中規模のワークロードでも意味がある場合があります。

PostgreSQL のマルチテナント SaaS パーティショニングモデル

マルチテナントを実現する最適な方法は、SaaS アプリケーションの要件によって異なります。以下のセクションでは、PostgreSQL にマルチテナント機能を正常に実装するためのパーティショニングモデルを示します。

> **Note**
>
> このセクションで説明するモデルは、Amazon RDS for PostgreSQL と Aurora PostgreSQL 互換の両方に適用できます。このセクションでの PostgreSQL への言及は、両方のサービスに適用されます。

PostgreSQL の SaaS パーティショニングに使用できる高レベルモデルには、サイロ、ブリッジ、プールの 3 つがあります。次の図は、サイロモデルとプールモデルのトレードオフをまとめたものです。ブリッジモデルは、サイロモデルとプールモデルのハイブリッドです。

パーティショニングモデル	利点	欠点
サイロ	• コンプライアンス調整 • テナント間の影響なし • テナントレベルのチューニング • テナントレベルのアベイラビリティ	• 俊敏性の低い • 一元化された管理なし • 導入の複雑さ • コスト
プール	• アジリティ • コスト最適化 • 一元化された管理 • 導入の簡略化	• テナント間の影響 • コンプライアンス上の課題 • オールオアナッシングアベイラビリティ
ブリッジ	• ある程度のコンプライアンス調整 • アジリティ	• コンプライアンスに関するいくつかの課題

パーティショニングモデル	利点	欠点
	• コスト最適化 • 一元化された管理	• すべてまたはまったくない可用性 (ほとんど) • テナント間の影響 • 導入の複雑さ

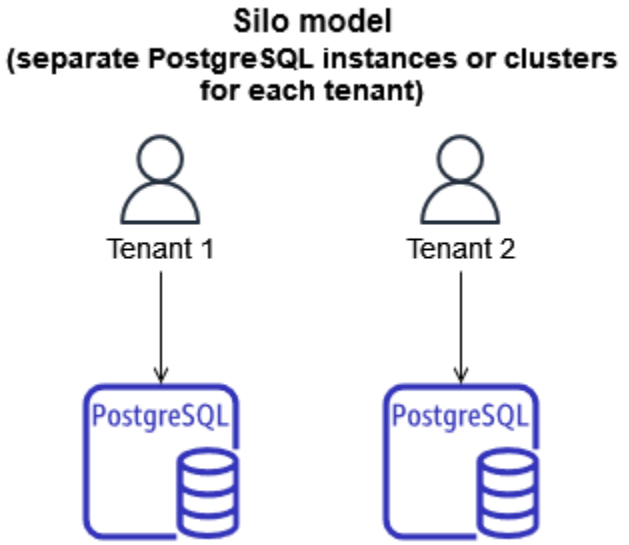
以下のセクションで、各モデルについて詳しく説明します。

パーティショニングモデル:

- PostgreSQL サイロモデル
- PostgreSQL プールモデル
- PostgreSQL ブリッジモデル
- 意思決定マトリックス

PostgreSQL サイロモデル

サイロモデルは、アプリケーション内の各テナントに PostgreSQL インスタンスをプロビジョニングすることによって実装されます。サイロモデルは、テナントのパフォーマンスとセキュリティの分離に優れており、ノイズの多いネイバー現象を完全に排除します。ノイジーネイバー現象は、あるテナントによるシステムの使用が別のテナントのパフォーマンスに影響する場合に発生します。サイロモデルでは、各テナントに合わせてパフォーマンスを調整でき、システム停止を特定のテナントのサイロに限定できる可能性があります。ただし、一般的にサイロモデルの採用を促進するのは、厳格なセキュリティと規制上の制約です。これらの制約は、SaaS のお客様によって動機付けられる可能性があります。たとえば、SaaSの顧客は内部の制約によりデータの分離を要求する場合があり、SaaSプロバイダーはそのようなサービスを追加料金で提供する場合があります。

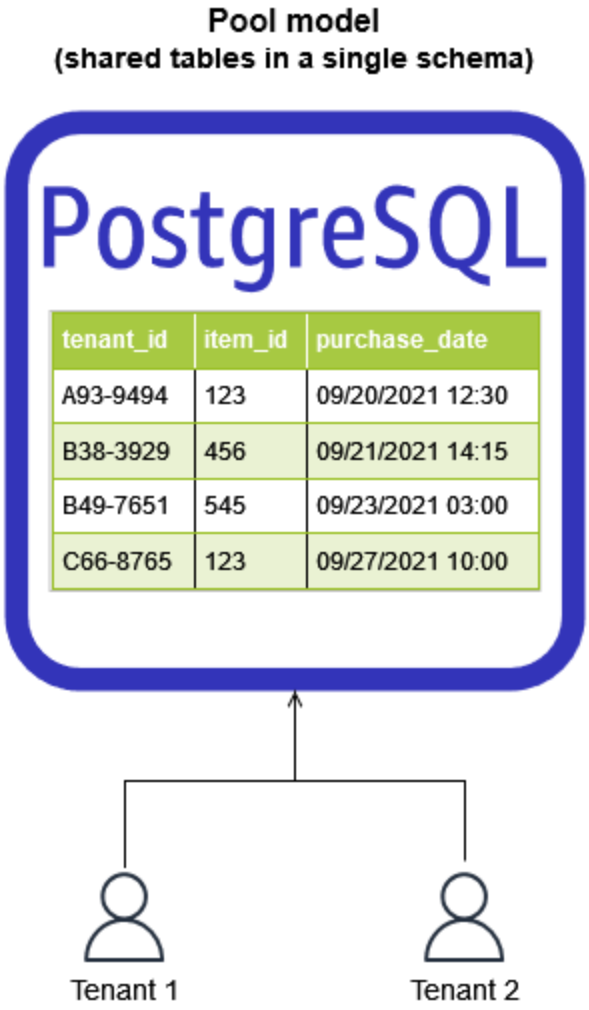


サイロモデルが必要になる場合もありますが、多くの欠点があります。複数のPostgreSQLインスタンスにわたるリソース消費の管理は複雑になる可能性があるため、サイロモデルを費用対効果の高い方法で使用することはしばしば困難です。さらに、このモデルではデータベースワークロードが分散しているため、テナントのアクティビティを一元的に把握することがより困難になります。独立して運用される多数のワークロードを管理すると、運用上および管理上のオーバーヘッドが増大します。また、サイロモデルでは、テナント固有のリソースをプロビジョニングする必要があるため、テナントのオンボーディングはより複雑で時間がかかります。さらに、テナント固有のPostgreSQLインスタンスの数が増え続けるにつれて、管理に必要な運用時間が長くなるため、SaaSシステム全体の拡張が困難になる可能性があります。最後に考慮すべき点は、アプリケーションまたはデータアクセスレイヤーがテナントとそれに関連するPostgreSQLインスタンスのマッピングを維持する必要があることです。これにより、このモデルの実装が複雑になります。

PostgreSQL プールモデル

プールモデルは、単一の PostgreSQL インスタンス (Amazon RDS または Aurora) をプロビジョニングし、行レベルのセキュリティ (RLS) を使用してテナントのデータを分離することで実装されます。RLS ポリシーは、SELECTクエリによってテーブル内のどの行が返されるか、またはどの行がINSERT、UPDATEDELETEおよびコマンドによって影響を受けるかを制限します。プールモデルでは、すべてのテナントデータが単一の PostgreSQL スキーマに一元化されるため、コスト効率が大幅に向上し、維持に必要な運用オーバーヘッドも少なくて済みます。また、このソリューションは集中管理されているため、監視も非常に簡単です。ただし、プールモデルにおけるテナント固有の影響を監視するには、通常、アプリケーションに追加のインストルメンテーションが必要です。これは、PostgreSQLはどのテナントがリソースを消費しているかをデフォルトで認識していないためで

す。新しいインフラストラクチャが不要なため、テナントのオンボーディングが簡単になります。この俊敏性により、テナントオンボーディングワークフローを迅速かつ自動化することが容易になります。

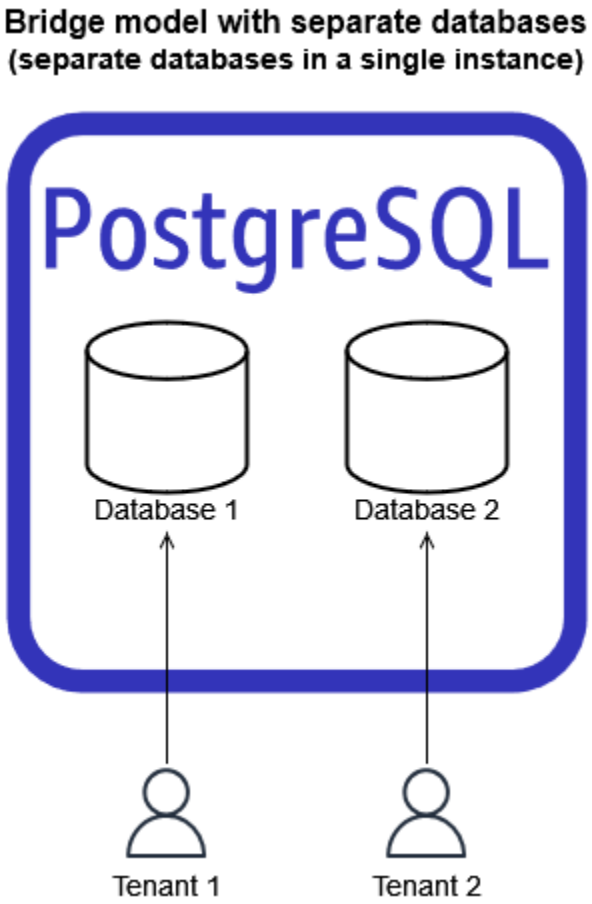


プールモデルは一般的に費用対効果が高く、管理も簡単ですが、いくつかの欠点もあります。プールモデルでは、ノイズの多い隣接現象を完全に排除することはできません。ただし、PostgreSQL インスタンスで適切なリソースを利用できるようにし、リードレプリカや Amazon へのクエリのオフロードなど、PostgreSQL ElastiCache の負荷を軽減する戦略を使用することで軽減できます。アプリケーションインストルメンテーションはテナント固有のアクティビティを記録および監視できるため、効果的な監視はテナントのパフォーマンス分離に関する懸念への対応にも役立ちます。最後に、一部のSaaS顧客は、RLSによる論理的な分離では十分ではなく、追加の分離手段を要求する場合があります。

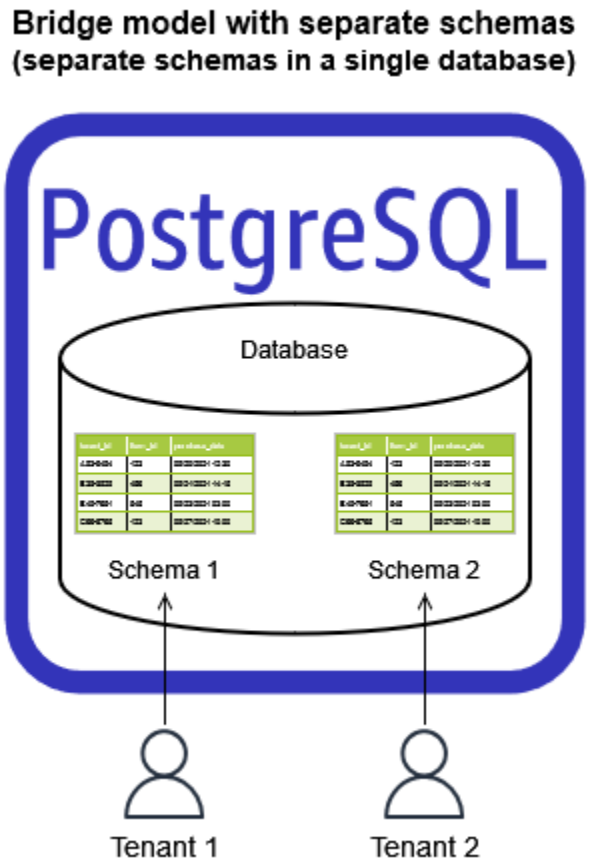
PostgreSQL ブリッジモデル

PostgreSQL ブリッジモデルは、プール型アプローチとサイロ型アプローチを組み合わせたものです。プールモデルと同様に、テナントごとに 1 つの PostgreSQL インスタンスをプロビジョニングします。テナントデータの分離を維持するには、PostgreSQL の論理構造を使用します。次の図では、PostgreSQL データベースを使用してデータを論理的に分離しています。

> **Note**
>
> PostgreSQL データベースは、個別の Amazon RDS for PostgreSQL または Aurora PostgreSQL と互換性のある DB インスタンスを参照しません。代わりに、データを分離するための PostgreSQL データベース管理システムの論理的な構成を指します。



次の図に示すように、各データベースにテナント固有のスキーマを含む単一の PostgreSQL データベースを使用してブリッジモデルを実装することもできます。



ブリッジモデルには、プールモデルと同じノイズの多いネイバーとテナントのパフォーマンスの分離という問題があります。また、テナントごとに個別のデータベースまたはスキーマをプロビジョニングする必要があるため、運用上およびプロビジョニング上のオーバーヘッドがいくらか発生します。テナントのパフォーマンスに関する懸念に迅速に対応するには、効果的な監視が必要です。また、テナント固有の使用状況を監視するためのアプリケーションインストルメンテーションも必要です。全体的に見て、ブリッジモデルは、新しい PostgreSQL データベースまたはスキーマが必要となるため、テナントのオンボーディング作業を少しだけ増やす RLS の代替手段と見なすことができます。サイロモデルと同様に、アプリケーションまたはデータアクセスレイヤーは、関連するPostgreSQL データベースまたはスキーマへのテナントのマッピングを維持する必要があります。

意思決定マトリックス

PostgreSQL で使用すべきマルチテナント SaaS パーティショニングモデルを決定するには、以下の決定マトリックスを参照してください。このマトリックスでは、次の 4 つのパーティショニングオプションが分析されます。

- サイロ — テナントごとに異なる PostgreSQL インスタンスまたはクラスタ。

- 個別のデータベースとのブリッジ — 単一の PostgreSQL インスタンスまたはクラスター内のテナントごとに個別のデータベース。
- 個別のスキーマによるブリッジ — 単一の PostgreSQL データベース、単一の PostgreSQL インスタンスまたはクラスタ内のテナントごとに個別のスキーマを作成します。
- プール — 単一インスタンスとスキーマ内のテナント用の共有テーブル。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
ユースケース	リソースの使用状況を完全に制御しながらデータを分離することが重要な要件です。また、テナントが非常に大規模でパフォーマンスに敏感な場合は、重要な要件となります。	データの分離は重要な要件であり、テナントのデータの相互参照は限られているか、まったく必要ありません。	中程度の数のテナントと中程度のデータ量。テナントのデータを相互参照する必要がある場合は、このモデルが適しています。	テナント数が多く、テナントあたりのデータ量が少ない。
新しいテナントのオンボーディングアジリティ	非常に遅い。(テナントごとに新しいインスタンスまたはクラスターが必要です)。	中中。中。中。中。(スキーマオブジェクトを格納するには、テナントごとに新しいデータベースを作成する必要があります)。	中中。中。中。中。(オブジェクトを保存するには、テナントごとに新しいスキーマを作成する必要があります)。	最速のオプション。(最低限の設定が必要です。)
データベース接続プール構成の労力と効率性	多大な労力が必要です。(テナントごとに 1 つの接続プール)	多大な労力が必要です。(Amazon RDS プロキシを使用し	必要な労力が少なくて済みます。(すべてのテ	必要な労力は最小限です。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
	効率が悪い。(テナント間でデータベース接続を共有することはできません。)	ない限り、テナントごとに 1 つの接続プール設定) 効率が悪い。(テナント間でのデータベース接続の共有や接続の総数はありません。 すべてのテナントでの使用量は、DB インスタンスクラスによって制限されます。)	ナントに 1 つの接続プール構成) 適度に効率的です。 (SET ROLESET SCHEMAまたはコマンドによる接続の再利用は、セッションプールモードでのみ。SETAmazon RDS Proxy を使用する場合、コマンドによってセッションが固定されることもありますが、クライアント接続プールは不要で、各リクエストに対して直接接続を行うことができるため、効率が向上します。)	最も効率的です。(1 つの接続プールですべてのテナントが使用でき、すべてのテナントで接続を効率的に再利用できます。データベース接続の制限は、DB インスタンスクラスによって異なります。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
データベースメンテナンス (バキューム管理) とリソース使用量	よりシンプルな管理。	中中中。中。中（後でデータベースごとにバキュームワーカーを起動する必要があるため、リソースの消費量が大きくなる可能性があります。これにより`vacuum_naptime` 、autovacuum Launcher の CPU 使用率が高くなります。 また、各データベースの PostgreSQL システムカタログテーブルのバキューム処理に関連する追加のオーバーヘッドが発生することもあります。)	大規模な PostgreSQL システムカタログテーブル。（テナント数とリレーション数によりますが、`pg_catalog` 合計サイズは 10 GB 単位です。 テーブルの膨張を抑えるには、バキューム関連のパラメーターの変更が必要になる可能性があります。)	テナント数とテナントごとのデータによっては、テーブルが大きくなる場合があります。(テーブルの膨張を抑制するには、バキューム関連のパラメーターの変更が必要になる可能性が高い)。
拡張機能管理の労力強化	多大な労力（データベースごとに別々のインスタンスで）。	多大な労力（各データベースレベルで）。	最小限の労力（共通データベースで 1 回）。	最小限の労力（共通データベースで 1 回）。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
導入労力を変更	労力労力を。(個々のインスタンスConnectし、変更をロールアウトします。)	労力労力を。(各データベースとスキーマConnectし、変更をロールアウトします。)	中中中。中。中(共通のデータベースConnectし、各スキーマの変更をロールアウトします。)	労力中。(共通データベースConnectし、変更をロールアウトします。)
導入の変更 — 影響範囲	最小限。(単一テナントが影響を受けます。)	最小限。(単一テナントが影響を受けます。)	最小限。(単一テナントが影響を受けます。)	非常に大きい。(すべてのテナントが影響を受けます。)
クエリパフォーマンスの管理と作業	管理可能なクエリパフォーマンス。	管理可能なクエリパフォーマンス。	管理可能なクエリパフォーマンス。	クエリのパフォーマンスを維持するには、多大な労力が必要になる可能性があります。(時間が経つにつれて、テーブルのサイズが大きくなるため、クエリの実行速度が遅くなる可能性があります。テーブルパーティショニングとデータベースシャーディングを使用してパフォーマンスを維持できます。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
テナント間のリソースへの影響	影響なし。(テナント間でのリソースの共有はありません。)	中中中中中中。(テナントはインスタンスの CPU やメモリなどの共通リソースを共有します。)	中中中中中中。(テナントはインスタンスの CPU やメモリなどの共通リソースを共有します。)	強いインパクト。(リソースやロックの競合などの観点から、テナント同士が相互に影響を及ぼします)。
テナントレベルのチューニング (テナントごとの追加インデックスの作成や特定のテナントの DB パラメーターの調整など)	可能な。	中。可能な、中。(スキーマレベルの変更はテナントごとに行うことができますが、データベースパラメータはすべてのテナントで共通です)。	中。可能な、中。(スキーマレベルの変更はテナントごとに行うことができますが、データベースパラメータはすべてのテナントで共通です)。	可能な、不可能な。(テーブルはすべてのテナントで共有されます。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
パフォーマンス重視のテナントのリバランス作業	最小限。（バランスを取り直す必要はありません。 このシナリオに対応するようにサーバーリソースと I/O リソースをスケーリングします。）	中中中中中（pg_dump論理レプリケーションまたはを使用してデータベースを エクスポートしますが、データサイズによってはダウンタイムが長くなる場合があります。 Amazon RDS for PostgreSQL のトランスポータブルデータベース機能を使用すると、インスタンス間でデータベースをすばやくコピーできます。）	中程度ですが、ダウンタイムが長くなる可能性があります。 (pg_dump論理レプリケーションまたはを使用してスキーマを エクスポートしますが、データサイズによってはダウンタイムが長くなる場合があります)。	すべてのテナントが同じテーブルを共有しているので重要です。(データベースをシャーディングするには、すべてを別のインスタンスにコピーし、テナントデータをクリーンアップするための追加手順が必要です)。 ほとんどの場合、アプリケーションロジックの変更が必要です。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
メジャーバージョンアップグレードによるデータベースのダウンタイム	標準ダウンタイム。(PostgreSQL システムカタログのサイズによって異なります。)	ダウンタイムが長くなる可能性があります。(システムカタログのサイズによって、時間は異なります。 PostgreSQL のシステムカタログテーブルもデータベース間で複製されます)	ダウンタイムが長くなる可能性があります。(PostgreSQL システムカタログのサイズによって、時間は異なります。)	標準ダウンタイム。(PostgreSQL システムカタログのサイズによって異なります。)
管理オーバーヘッド (データベースログ分析やバックアップジョブの監視など)	労力労力を	労力中。	労力中。	労力中。
テナントレベルのアベイラビリティ	最大。(各テナントは個別に障害が発生し、回復します。)	影響範囲が広い。(ハードウェアまたはリソースの問題が発生すると、すべてのテナントで障害が発生し、同時に復旧します。)	影響範囲が広い。(ハードウェアまたはリソースの問題が発生すると、すべてのテナントで障害が発生し、同時に復旧します。)	影響範囲が広い。(ハードウェアまたはリソースの問題が発生すると、すべてのテナントで障害が発生し、同時に復旧します。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
テナントレベルのバックアップとリカバリの作業	労力中。(テナントごとにバックアップと復元が可能です。)	中中中。中。中 (テナントごとに論理的なエクスポートとインポートを使用してください。ある程度のコーディングと自動化が必要です。)	中中中。中。中 (テナントごとに論理的なエクスポートとインポートを使用してください。ある程度のコーディングと自動化が必要です。)	労力労力を。(すべてのテナントが同じテーブルを共有します。)
point-in-time テナントレベルの復旧作業	労力中。(スナップショットを使用してポイントインタイムリカバリを使用するか、Amazon Aurora でバックトラッキングを使用してください)。	中中中。中。中 (スナップショットのリストアを使用してから、エクスポート/インポートを行います。 ただし、これは処理に時間がかかります。)	中中中。中。中 (スナップショットのリストアを使用してから、エクスポート/インポートを行います。 ただし、これは処理に時間がかかります。)	多大な労力と複雑さ。
統一スキーマ名	テナントごとに同じスキーマ名。	テナントごとに同じスキーマ名。	テナントごとに異なるスキーマ。	共通スキーマ。
テナントごとのカスタマイズ (特定のテナントのテーブル列の追加など)	可能な。	可能な。	可能な。	複雑 (すべてのテナントが同じテーブルを共有しているため)。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
オブジェクトリレーショナルマッピング (ORM) レイヤー (Ruby など) でのカタログ管理の効率性	効率的 (クライアント接続はテナントごとに異なるため)。	効率的 (クライアント接続はデータベースに固有であるため)。	適度に効率的です。(使用する ORM、ユーザー/ロールのセキュリティモデル、`search_path` および構成によっては、クライアントはすべてのテナントのメタデータをキャッシュすることがあり、DB 接続のメモリ使用量が多くなります)。	効率的 (すべてのテナントが同じテーブルを共有するため)。
テナント報告の統合作業	労力労力を。(すべてのテナントのデータを統合するか、[ETL] を別のレポートデータベースに抽出、変換、ロードするには、外部データラッパー [FDW] を使用する必要があります)。	労力労力を。(すべてのテナントまたはETLのデータを別のレポートデータベースに統合するには、FDWを使用する必要があります)。	中中中。中。中 (ユニオンを使用すると、すべてのスキーマのデータを集約できます)。	労力中。(すべてのテナントデータは同じテーブルにあるため、レポートは簡単です。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
レポート用のテナント固有の読み取り専用インスタンス (サブスクリプションベースなど)	労力中。(リードレプリカを作成します。)	中中中。中。中 (AWS論理レプリケーションまたはDatabase Migration Service [AWS DMS] を使用して構成できます。)	中中中。中。中 (論理レプリケーションを使用することもAWS DMS、構成することもできます)。	複雑 (すべてのテナントが同じテーブルを共有しているため)。
データ分離	最高。	ベスト。(データベースレベルの権限はPostgreSQLロールを使用して管理できます。)	ベスト。(PostgreSQL ロールを使用してスキーマレベルの権限を管理できます。)	さらに悪い。(すべてのテナントが同じテーブルを共有するため、テナントを分離するための行レベルのセキュリティ [RLS] などの機能を実装する必要があります)。
テナント固有のストレージ暗号化キー	可能な。(各PostgreSQLクラスターには、ストレージ暗号化用の独自のAWS Key Management Service [AWS KMS] キーを設定できます。)	可能な、不可能な。(すべてのテナントは、ストレージ暗号化用の同じ KMS キーを共有します。)	可能な、不可能な。(すべてのテナントは、ストレージ暗号化用の同じ KMS キーを共有します。)	可能な、不可能な。(すべてのテナントは、ストレージ暗号化用の同じ KMS キーを共有します。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
各テナントのデータベース認証に AWS Identity and Access Management (IAM) を使用する	可能な。	可能な。	可能 (各スキーマに別々の PostgreSQL ユーザーを割り当てることにより)。	できません (テーブルはすべてのテナントで共有されているため)。
インフラストラクチャコスト	最高 (何も共有されていないため)。	中中中中中	中中中中中	最中。
データの複製とストレージの使用	全テナントで最も高い集計。(PostgreSQL システムカタログテーブルとアプリケーションの静的データと共通データは、すべてのテナントで重複しています。)	全テナントで最も高い集計。(PostgreSQL システムカタログテーブルとアプリケーションの静的データと共通データは、すべてのテナントで重複しています。)	中中中中中 (アプリケーションの静的データと共通データは、共通のスキーマに格納されていて、他のテナントからもアクセスできます)。	最小限。(データの重複はありません。 アプリケーションの静的データと共通データは同じスキーマにあってもかまいません。)

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
テナント中心の監視 (問題の原因となっているテナントを迅速に特定)	労力中。(各テナントは個別に監視されるため、特定のテナントのアクティビティを簡単に確認できます。)	中中中。中。中 (すべてのテナントが同じ物理リソースを共有するため、特定のテナントのアクティビティを確認するには追加のフィルタリングを適用する必要があります)。	中中中。中。中 (すべてのテナントが同じ物理リソースを共有するため、特定のテナントのアクティビティを確認するには追加のフィルタリングを適用する必要があります)。	多大な労力が必要です。(すべてのテナントはテーブルを含むすべてのリソースを共有するため、バインド変数キャプチャを使用して特定の SQL クエリがどのテナントに属しているかを確認する必要があります)。
一元管理とヘルス/アクティビティモニタリング	多大な労力 (中央監視と中央コマンドセンターの設置) 。	中程度の労力 (すべてのテナントが同じインスタンスを共有するため)。	中程度の労力 (すべてのテナントが同じインスタンスを共有するため)。	最小限の労力 (すべてのテナントがスキーマを含む同じリソースを共有するため)。

	サイロ	個別のデータベースとのブリッジ	個別のスキーマを使ったブリッジ	プール
オブジェクト識別子 (OID) とトランザクション ID (XID) のラップアラウンドが発生する可能性	最小限。	高。高。（OID、XIDはPostgreSQLクラスタ全体の単一カウンターであるため、物理データベース間で効果的にバキューム処理を行うと問題が発生する可能性があります）。	中中中中中 (OID、XID は PostgreSQL クラスタ全体にわたる単一カウンタであるため)。	高。高。（たとえば、 out-of-line 列数によっては、1つのテーブルでTOAST OIDの上限である40億に達することがあります。）

行レベルのセキュリティ

PostgreSQL のプールモデルでテナントデータの分離を維持するには、行レベルのセキュリティ (RLS) が必要です。RLS は、データベースレベルでの隔離ポリシーの適用を一元的に行い、ソフトウェア開発者からの隔離を維持する負担を軽減します。RLS を実装する最も一般的な方法は、PostgreSQL DBMS でこの機能を有効にすることです。RLS では、指定された列の値に基づいてデータ行へのアクセスをフィルタリングします。データへのアクセスをフィルタリングするには、次の 2 つの方法を使用できます。

- テーブル内の指定されたデータ列は、現在の PostgreSQL ユーザーの値と比較されます。ログインしている PostgreSQL ユーザーと同等の列の値には、そのユーザーがアクセスできます。
- テーブル内の指定されたデータ列は、アプリケーションによって設定された実行時変数の値と比較されます。実行時変数と同等の列の値には、そのセッション中にアクセスできます。

最初のオプションではテナントごとに新しい PostgreSQL ユーザーを作成する必要があるため、2 番目のオプションが推奨されます。代わりに、PostgreSQL を使用する SaaS アプリケーションが PostgreSQL に問い合わせるときに、実行時にテナント固有のコンテキストを設定する必要があります。これは RLS を強制する効果があります。RLS table-by-table は個別に有効にすることもできます。ベストプラクティスとして、テナントデータを含むすべてのテーブルでRLSを有効にするようにしてください。

次の例では 2 つのテーブルを作成し RLS をクリックします。この例では、`app.current_tenant`データ列を実行時変数の値と比較します。

```
-- Create a table for our tenants with indexes on the primary key and the tenant’s name
CREATE TABLE tenant (
    tenant_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    name VARCHAR(255) UNIQUE,
    status VARCHAR(64) CHECK (status IN ('active', 'suspended', 'disabled')),
    tier VARCHAR(64) CHECK (tier IN ('gold', 'silver', 'bronze'))
);

-- Create a table for users of a tenant
CREATE TABLE tenant_user (
    user_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    tenant_id UUID NOT NULL REFERENCES tenant (tenant_id) ON DELETE RESTRICT,
    email VARCHAR(255) NOT NULL UNIQUE,
    given_name VARCHAR(255) NOT NULL CHECK (given_name <> ''),
    family_name VARCHAR(255) NOT NULL CHECK (family_name <> '')
```

```
);

-- Turn on RLS
ALTER TABLE tenant ENABLE ROW LEVEL SECURITY;

-- Restrict read and write actions so tenants can only see their rows
-- Cast the UUID value in tenant_id to match the type current_setting
-- This policy implies a WITH CHECK that matches the USING clause
CREATE POLICY tenant_isolation_policy ON tenant
USING (tenant_id = current_setting('app.current_tenant')::UUID);

-- And do the same for the tenant users
ALTER TABLE tenant_user ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_user_isolation_policy ON tenant_user
USING (tenant_id = current_setting('app.current_tenant')::UUID);
```

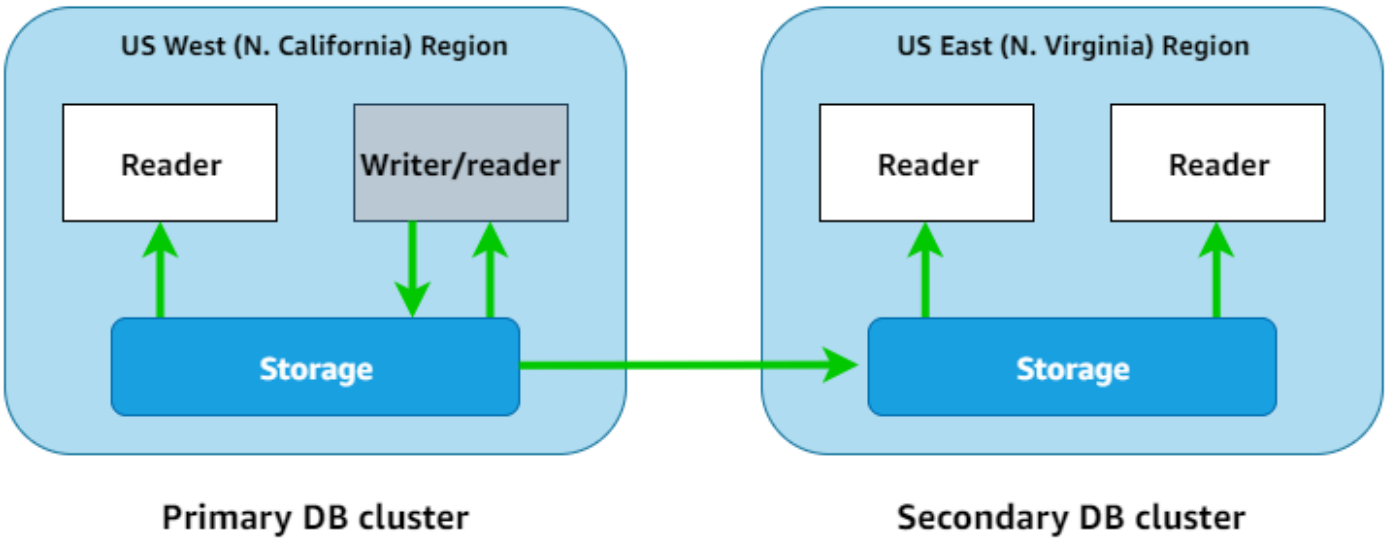
詳細については、ブログ記事「PostgreSQL 行レベルのセキュリティ) をクリックします。AWSSaaS Factoryチームには、RLSの実装に役立ついくつかの例もあります。 GitHub

プールモデルの PostgreSQL の可用性

プールモデルの性質上、PostgreSQL インスタンスは 1 つだけです。したがって、高可用性を実現するためにアプリケーションを設計することが重要です。プールされたデータベースの障害または停止により、アプリケーションがすべてのテナントで低下したり、アクセスできなくなったりします。

Amazon RDS for PostgreSQL DB インスタンスは、高可用性機能を有効にすることで、2 つのアベイラビリティーゾーン間で冗長化できます。詳細については、Amazon RDS ドキュメントの「Amazon RDS の高可用性 (マルチ AZ)」を参照してください。クロスリージョンフェイルオーバーでは、別の AWS リージョンにリードレプリカを作成できます。(このリードレプリカは、フェイルオーバープロセスの一部として昇格する必要があります)。さらに、 AWS リージョン間でレプリケートされたバックアップをレプリケートして復旧することもできます。詳細については、Amazon RDS ドキュメントの「自動バックアップを別の AWS リージョンにレプリケートする」を参照してください。

Aurora PostgreSQL - 互換 は、複数のアベイラビリティーゾーンの障害を維持できる方法でデータを自動的にバックアップします。(Aurora ドキュメントの「Amazon Aurora の高可用性」を参照してください。) Aurora の回復力を高め、復旧を高速化するために、他のアベイラビリティーゾーンに Aurora リードレプリカを作成できます。Aurora グローバルデータベースを使用して、クロス AWS リージョンリカバリと自動フェイルオーバーのために、データを 5 つの追加リージョンにレプリケートできます。(Aurora ドキュメントの「Amazon Aurora グローバルデータベースの使用」を参照してください。) さらに、Aurora グローバルデータベースで書き込み転送を有効にして、複数の で高可用性を実現できます AWS リージョン。



Amazon RDS for PostgreSQL と Aurora PostgreSQL 互換のどちらを使用しているかにかかわらず、プールモデルを使用するすべてのマルチテナント SaaS アプリケーションの停止の影響を軽減するために、高可用性機能を実装することをお勧めします。

ベストプラクティス

このセクションでは、このガイドの要点をいくつか紹介します。各点の詳細については、該当するセクションへのリンクを参照してください。

マネージド PostgreSQLAWS のオプションを比較

AWSには、マネージド環境で PostgreSQL を実行する主な方法が 2 つあります。(ここでいうマネージドとは、PostgreSQL インフラストラクチャと DBMSAWS がサービスによって部分的または完全にサポートされていることを意味します。) AWSマネージドPostgreSQLオプションには、PostgreSQLのバックアップ、フェールオーバー、最適化、および一部の管理を自動化できるという利点があります。マネージドオプションとして、Amazon Aurora PostgreSQL 互換エディションと PostgreSQL 向けの Amazon Relational Database Service (Amazon RDS)AWS を提供しています。PostgreSQL のユースケースを分析することで、これら 2 つのモデルから最適なモデルを選択できます。詳細については、このガイドの「Amazon RDS と Aurora のどちらを選択する」セクションを参照してください。

マルチテナント SaaS パーティショニングモデルを選択

PostgreSQLに適用できる3つのSaaSパーティショニングモデル (サイロ、ブリッジ、プール) から選択できます。各モデルには長所と短所があり、ユースケースに応じて最適なモデルを選択する必要があります。Amazon RDS for PostgreSQL と Aurora PostgreSQL 互換は 3 つのモデルすべてをサポートしています。SaaS アプリケーションのテナントデータの分離を維持するには、モデルを選択することが重要です。これらのモデルの詳細については、本ガイドの「PostgreSQL のマルチテナント SaaS パーティショニングモデル」セクションを参照してください。

プール SaaS パーティショニングモデルに行レベルのセキュリティを使用する

PostgreSQL を使用するプールモデルでテナントデータの分離を維持するには、行レベルのセキュリティ (RLS) が必要です。これは、プールモデルでは、インフラストラクチャ、PostgreSQL データベース、またはスキーマをテナントごとに論理的に分離していないためです。RLS は、データベースレベルでの隔離ポリシーの適用を一元的に行い、ソフトウェア開発者からの隔離を維持する負担を軽減します。RLS を使用して、データベース操作を特定のテナントに制限できます。詳細と例については、本ガイドの「行レベルのセキュリティに関する推奨事項」セクションを参照してください。

よくある質問

このセクションでは、マネージド PostgreSQL をマルチテナントの SaaS アプリケーションに実装することについてよく寄せられる質問に対する回答を提供します。

どのマネージド PostgreSQLAWS オプションが提供されていますか?

AWSは、PostgreSQL 向けの Amazon Aurora PostgreSQL 互換および Amazon Relational Database Service (Amazon RDS) を提供しています。 AWSには、マネージドデータベース製品の幅広いカタログもあります。

SaaS アプリケーションに最適なサービスはどれですか?

SaaS アプリケーションには Aurora PostgreSQL 互換と Amazon RDS for PostgreSQL の両方を使用できます。また、このガイドで説明されているすべての SaaS パーティショニングモデルも使用できます。これら 2 つのサービスには、スケーラビリティ、クラッシュリカバリ、フェイルオーバー、ストレージオプション、高可用性、ディザスタリカバリ、バックアップ、および各オプションで使用できるインスタンスクラスが異なります。最適な選択肢は、特定のユースケースに大きく左右されます。このガイドの意思決定マトリックスを使用して、ユースケースに最適なオプションを選択してください。

マルチテナント SaaS アプリケーションで PostgreSQL データベースを使用する場合、考慮すべき固有の要件はどれですか?

SaaS アプリケーションで使用される他のデータストアと同様に、最も重要な考慮事項はテナントデータの分離を維持する方法です。このガイドで説明したように、AWSマネージドPostgreSQLサービスを使用してテナントデータを分離する方法は複数あります。さらに、どのPostgreSQL実装でも、テナントごとにパフォーマンスを分離することを検討する必要があります。

PostgreSQL でテナントのデータ分離を維持するにはどのモデルを使用できますか?

サイロ、ブリッジ、プールモデルを SaaS パーティショニング戦略として使用して、テナントデータの分離を維持できます。これらのモデルと PostgreSQL への適用方法についての説明については、本

ガイドの「PostgreSQL のマルチテナント SaaS パーティショニングモデル」セクションを参照してください。

複数のテナントで共有されている単一の PostgreSQL データベースでテナントのデータを分離する方法を教えてください。

PostgreSQL は、単一の PostgreSQL データベースまたはインスタンスでテナントデータを分離するために使用できる行レベルセキュリティ (RLS) 機能をサポートしています。さらに、この目標を達成するために、テナントごとに個別の PostgreSQL データベースを 1 つのインスタンスでプロビジョニングすることも、テナントごとにスキーマを作成することもできます。これらのアプローチの長所と短所については、本ガイドの「行レベルのセキュリティに関する推奨事項」セクションを参照してください。

次のステップ

AWSには、マネージド PostgreSQL の運用方法として Aurora PostgreSQL 互換と Amazon RDS for PostgreSQL の 2 つのオプションが用意されています。2 つのサービスを評価し、マルチテナント SaaS アプリケーションの特定のユースケースに最も適したオプションを選択することをお勧めします。SaaS パーティショニングモデルに準拠することで、PostgreSQL を使用する SaaS アプリケーションが、テナンシーを維持するためのベストプラクティスに厳密に従っていることを確認できます。SaaS サイロ、ブリッジ、プールパーティショニングモデルは、多くの SaaS ユースケースをサポートします。これらのモデルには、パフォーマンスの分離、運用上のオーバーヘッド、テナントのセキュリティなどの要因によってさまざまな利点があります。

次のステップ:

- Aurora PostgreSQL 互換および Amazon RDS for PostgreSQL を評価し、お客様の SaaS アプリケーションに最適なオプションを選択してください。
- アプリケーションの要件を満たす SaaS パーティショニングモデル (サイロ、ブリッジ、プール) を選択してください。
- 選択したSaaSパーティショニングモデルに従ってPostgreSQLを実装します。

リソース

リファレンス

- SaaS ストレージ戦略: でのマルチテナントストレージモデルの構築 AWS (AWS ホワイトペーパー)
- Amazon Aurora PostgreSQL 用 Amazon Aurora Global Database を使用したクロスリージョンディザスタリカバリ (AWS ブログ記事)
- PostgreSQL 行レベルセキュリティによるマルチテナントデータ分離 (AWS ブログ記事)
- 「Amazon Aurora PostgreSQL の操作」 (Aurora ドキュメント)
- Amazon RDS の PostgreSQL (Amazon RDS ドキュメント)

パートナー

- Amazon Aurora for PostgreSQL パートナー
- Amazon RDS for PostgreSQL パートナー

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
更新	Aurora での書き込み転送の可用性を反映するように更新されました。	2024 年 4 月 29 日
更新	Amazon RDS と Aurora 比較テーブル を更新しました。	2022 年 10 月 21 日
—	初版発行	2021 年 9 月 30 日

AWS 規範的ガイダンスの用語集

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 PostgreSQL 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の最適化レベルを導入します。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: カスタマーリレーションシップ管理 (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 インスタンス上の 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: Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。

- 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

「アトミック性、一貫性、分離性、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。アクティブ/パッシブ移行よりも柔軟ですが、より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

行のグループを操作し、グループの単一の戻り値を計算する SQL 関数。集計関数の例としては、 SUMや などがあります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。

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じリージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドに正常に移行 AWS するための効率的で効果的な計画を立てるのに役立つ、 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。 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編成します。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーション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 のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool （ AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人または組織に混乱や損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 の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントの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。エンディアンネス 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 つの異なる同一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、影響を最小限に抑えながら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のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボット と呼ばれる他のボッ

トの中には、個人や組織に混乱を与えたり、損害を与えたりすることを意図しているものがあります。

ボットネット

マルウェアに感染し、ボットのヘルダーまたはボットオペレーター と呼ばれる、単一関係者の管理下にあるボットのネットワーク。ボットは、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

例外的な状況や承認されたプロセスを通じて、ユーザーが通常アクセス許可を持たない AWS アカウント にすばやくアクセスできるようにします。詳細については、 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。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

AWS 「クラウド導入フレームワーク」を参照してください。

Canary デプロイ

エンドユーザーへのバージョンの低速かつ増分的なリリース。確信できたら、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。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストします。AWS Fault Injection Service （ AWS FIS ） を使用して、 AWS ワークロードに負荷をかけてレスポンスを評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

クライアント側の暗号化

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

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 の投稿」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

組織が に移行するときに通常実行する 4 つのフェーズ 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
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略とどのように関連しているかについては、 AWS 「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、 GitHub または が含まれます AWS CodeCommit。コードの各バージョンはブランチと呼ばれます。マイクロサー

ビスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオ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、設定は想定した状態から変化します。これにより、ワークロードが非準拠になる可能性があり、通常は段階的かつ意図的ではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンで単一のエンティティとしてデプロイすることも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性

の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、データ分類を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

一元化された管理とガバナンスにより、分散型の分散型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。信頼できる ID のみが、期待されるネットワークから信頼できるリソースにアクセスしていることを確認できます。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには通常、大量の履歴データが含まれており、クエリや分析によく使用されます。

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 に採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。例えば、 defense-in-depth アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS Organizations ドキュメントのAWS Organizationsで使用できるサービスを参照してください。

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、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマ 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数値です。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、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差の追跡。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 ガバナンス要件への準拠に影響を与える可能性のあるランディングゾーンの変更を検出したりできます。

DVSM

「開発値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、応答時間を短縮できます。

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) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (アカウンティング、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service （ AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。
- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイド を参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

スタースキーマ の中央テーブル。事業運営に関する定量的データを保存します。通常、ファクトテーブルには、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の 2 種類の列が含まれます。

フェイルファスト

開発ライフサイクルを短縮するために頻繁で段階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、障害の影響を制限し AWS リージョン、ワークロードの耐障害性を向上させるアベイラビリティーゾーン、コントロールプレーン、データプレーンなどの境界です。詳細については、AWS 「障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性：AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

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制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

段階的なアプローチを使用するのではなく、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、可能な限り短い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ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは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推奨されるアプローチです。

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

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装

されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは、 AWS Config、、Amazon AWS Security Hub、GuardDuty、Amazon Inspector AWS Trusted Advisor、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

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性のため、通常、修正は一般的な DevOps リリースワークフローの外で行われます。

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 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「産業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、または変更するのではなく、本番ワークロード用の新しいインフラストラクチャをデプロイするモデル。イミュータブルなインフラストラクチャは、本質的にミュータブルなインフラストラクチャ よりも一貫性、信頼性、予測性が高くなります。詳細については、 AWS Well-Architected フレームワークの「イミュータブルインフラストラクチャを使用したデプロイ」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション外からのネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリ

ケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

接続、リアルタイムデータ、自動化、分析、AI/ML の進歩を通じて、 のビジネス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。

産業分野における IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「Building an industrial Internet of Things (IIoT) digital transformation strategy」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「AWS を使用した機械学習モデルの解釈」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 ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイド を参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロー

ドとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、安全でスケーラブルなマルチアカウント AWS 環境のセットアップ を参照してください。

大規模な移行

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特権アクセス許可を適用するを参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。エンディアンネス 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ているソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスにつながる

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 がインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、ユーザーがエンドポイントにアクセスしてデータを保存および取得します。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、 マネージドサービスの例です。これらは抽象化されたサービス とも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するためのソフトウェアシステム。これにより、加工品を現場の完成製品に変換します。

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、ツールの導入を推進し、調整のために結果を検査する完全なプロセス。メカニズムは、動作中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 デバイス用の、パブリッシュ/サブスクライブパターンに基づく軽量の machine-to-machine (M2M) 通信プロトコル。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロ

イ、再利用可能なコード、回復力などがあります。詳細については、AWS 「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

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。組織がクラウドへの移行のための強固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ちます。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、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、スプリントに取り組む 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: Application Migration Service を使用して Amazon EC2 AWS への移行をリホストします。

Migration Portfolio Assessment (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 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイド を参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを に移行するために使用されるアプローチ AWS クラウド。詳細については、この用語集の「7 Rs エントリ」と「組織を動員して大規模な移行を加速する」を参照してください。

ML

「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「」の「アプリケーションをモダナイズするための戦略 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。

変更可能な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 AWS Framework では、一貫性、信頼性、予測可能性を向上させるために、イミュータブルなインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセスコントロール」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。

オープンプロセス通信 - 統合アーキテクチャ (OPC-UA)

産業用オートメーション用の machine-to-machine (M2M) 通信プロトコル。OPC-UA は、データの暗号化、認証、認可スキームを備えた相互運用性標準を提供します。

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

インシデントや潜在的な障害の理解、評価、防止、または範囲の縮小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「運用準備状況レビュー (ORR)」を参照してください。

運用テクノロジー (OT)

産業運用、機器、インフラストラクチャを制御するために物理環境と連携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 トランスフォーメーションの主要な焦点です。

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

の組織 AWS アカウント 内のすべての のすべてのイベントを口グ AWS CloudTrail 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、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードから、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 のすべての 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、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のアクセスを制限するオプションです。OAI を使用すると、 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。OACも併せて参照してください。OAC では、より詳細な、強化されたアクセスコントロールが可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用技術」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されるネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

「個人を特定できる情報」を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラ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。`false`通常は `WHERE`句にあります。

述語のプッシュダウン

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に関する用語と概念内にあるプリンシパルを参照してください。

プライバシーバイデザイン

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

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠のリソースのデプロイを防止する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まで、製品のデータとプロセスのライフサイクル全体にわたる管理、および辞退と削除。

本番環境

「環境」を参照してください。

プログラミング可能ロジックコントローラー (NAL)

製造では、マシンをモニタリングし、承認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。

パブリッシュ/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES では、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発行できます。システムは、公開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され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、通知 (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、通知 (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。

再構築

[「7 Rs」を参照してください]。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

[「7 Rs」を参照してください]。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の とは分離され、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 R」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

中断に耐えたり、中断から回復したりするアプリケーションの機能。で障害耐性を計画する場合、高可用性とディザスタリカバリが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。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任

(A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートを含めると、そのマトリックスは RASCI マトリックスと呼ばれ、サポートを除外すると 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 R」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。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

多くの ID プロバイダー (IdPs) が使用するオープンスタンダード。この機能により、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは にログイン AWS

Management Console したり AWS 、 API オペレーションを呼び出したりできます。組織内のすべてのユーザーに対して IAM でユーザーを作成する必要はありません。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントのSAML 2.0 ベースのフェデレーションについてを参照してください。

SCADA

「監視コントロールとデータ収集」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、単一の文字列、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 、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。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントに自動的に応答または修正するように設計された、事前定義されプログラムされたアクション。これらのオートメーションは、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 インスタンスへのパッチ適用、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

送信先にあるデータの、それを受け取る AWS のサービス による暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、AWS 全般のリファレンスの「AWS のサービス エンドポイント」を参照してください。

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットなど、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)

サービスレベルのインジケータ によって測定される、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、責任共有モデルを参照してください。

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)

システムを中断す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

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

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「アプリケーションをモダナイズするための段階的アプローチ 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。

star スキーマ

トランザクションデータまたは測定データを保存するために 1 つの大きなファクトテーブルを使用し、データ属性を保存するために 1 つ以上の小さなディメンションテーブルを使用するデータベースの組織構造。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の目的で使用するように設計されています。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズを参照してください。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視統制とデータ収集 (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生産オペレーションをモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザー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

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定するサービスへの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AWS Organizations の「 を他の AWS のサービスで使用する 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できる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに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 1 回」を参照し、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。

Write Once, Read Many (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。承認されたユーザーは、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な 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 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。