Amazon Neptune データベースを実行する ISVs のマルチテナンシーガイダンス

AWS 規範ガイダンス

AWS 規範ガイダンス: Amazon Neptune データベースを実行する ISVs のマルチテナンシーガイダンス

Table of Contents

Amazon Neptune データベースを実行する ISVs のマルチテナンシーガイダンス

Amazon Web Services (寄稿者)

2024 年 8 月 (ドキュメント履歴)

マルチテナンシーは、アプリケーションの単一のインスタンスが複数のお客様にサービスを提供するコンピュータシステムアーキテクチャです。各顧客はテナントと呼ばれます。マルチテナントアーキテクチャでは、アプリケーションのこれらのインスタンスは、各テナントが同じインフラストラクチャに物理的に配置され、論理的に分離されている共有環境で動作します。

独立系ソフトウェアベンダー (ISV) として、Amazon Neptune を使用して、高度に接続されたデータ間のナビゲーションを必要とするアプリケーションを強化できます。アカウントでクラウドベースの Software as a Service (SaaS) アプリケーションを管理し、テナントにサブスクリプションを提供する場合があります。その後、テナントはインターネット経由で、または 経由でプライベートにサービスにアクセスできます AWS PrivateLink。このモデルの経済性は、テナントが購入、構築、保守するよりも安価なソフトウェアにアクセスできるため、両者にとって有効です。ISV として、サブスクリプションに対して、ソフトウェアの作成と保守にかかるコストよりも多くの料金を請求できます。問題は、ビジネスを複数のテナントにスケールする方法です。

マルチテナンシーはISVsに重要な経済的および運用上の利点を提供します。マルチテナントアーキテクチャにより、組織は投資収益率 (ROI) が向上します。マルチテナンシーは運用要件を簡素化し、組織がより迅速に移行し、テナントにソフトウェアを配信するコストを削減できるようにします。

このドキュメントでは、Amazon Neptune を使用してマルチテナント ISV アプリケーションを効果的に実行するためのガイダンスを提供します。このガイダンスは、ISVs による顧客への SaaS ソリューションの正常な提供を長年にわたってサポートしてきたベストプラクティスに基づいています。組織の目標とアーキテクチャ原則に照らしてこのガイダンスを評価することは、ソリューションを最適化する方法を見つけるのに役立ちます。

> **Note**
>
> このドキュメントでは、ベストプラクティスを網羅したリストは提供していません。マルチテナンシー ISV ワークロードに関する追加の具体的なガイダンスを提供することで、ドキュメントAmazon Neptune の AWS Well-Architected フレームワークの適用」を補足します。ソ

リューションを設計するときは、両方のドキュメントの考慮事項を確認することをお勧めします。

SaaS データパーティショニングモデル

SaaS デベロッパーにとっての課題の 1 つは、マルチテナント環境でデータを表現および整理するためのアーキテクチャパターンを設計することです。これらのマルチテナントストレージのメカニズムとパターンは、通常、データパーティショニング と呼ばれます。

マルチテナント SaaS 環境では、データパーティショニングとテナント分離を区別することが重要です。これらの概念は関連していますが、同義語ではありません。データパーティショニングとは、各テナントのデータを保存する方法を指します。ただし、パーティショニングだけではテナントの分離が保証されません。あるテナントのデータが別のテナントにアクセスできないようにするには、追加の対策が必要です。

マルチテナント SaaS システムの 3 つの一般的なデータパーティショニングモデルは、サイロ、プール、ハイブリッドです。どのモデルを選択するかは、次のような要因によって異なります。

- コンプライアンス
- ノイズの多い隣人
- 階層化戦略
- 運用要件
- テナント分離のニーズ

さらに、 で使用できる各データベースタイプには、 AWS 通常、データパーティショニングとテナント分離モデルの一意のコレクションが用意されています。ソリューションのさまざまなニーズをサポートするためにテナントグラフを整理する方法を検討するときは、Amazon Neptune が提供するモデルを検討してください。

多くの場合、次のいずれかのアサーションを使用して Neptune で設計ISVsを開始します。

- このISVソリューションでは、別々のクラスター間で顧客を物理的に分離する必要があります。
- このISVソリューションには、従来のリレーショナルデータベース管理システムにある名前付きデータベースやスキーマなどの構築が必要です。

検討後、これらのアサーションは正しくないISVsことを認識してください。これは、ほぼすべてのワークロードで、各顧客がデータベースに切断されたグラフを持っているためです。このドキュメントで説明されているデータモデリングとアクセスガイダンスを実装すると、これらのデータ境界が交差し、顧客のデータプライバシーが維持されます。

このガイドでは、サイロモデル とプールモデル の両方について説明しますが、ほとんどの場合、コストと運用効率のためにプールモデルISVsを選択します。このガイドでは、サイロモデルとプールモデルの両方の側面を組み合わせたハイブリッドモデルについて簡単に説明します。グラフサイズの規制またはコンプライアンス要件に対応するために、最大の顧客向けにハイブリッドモデルISVsを使用する人もいます。

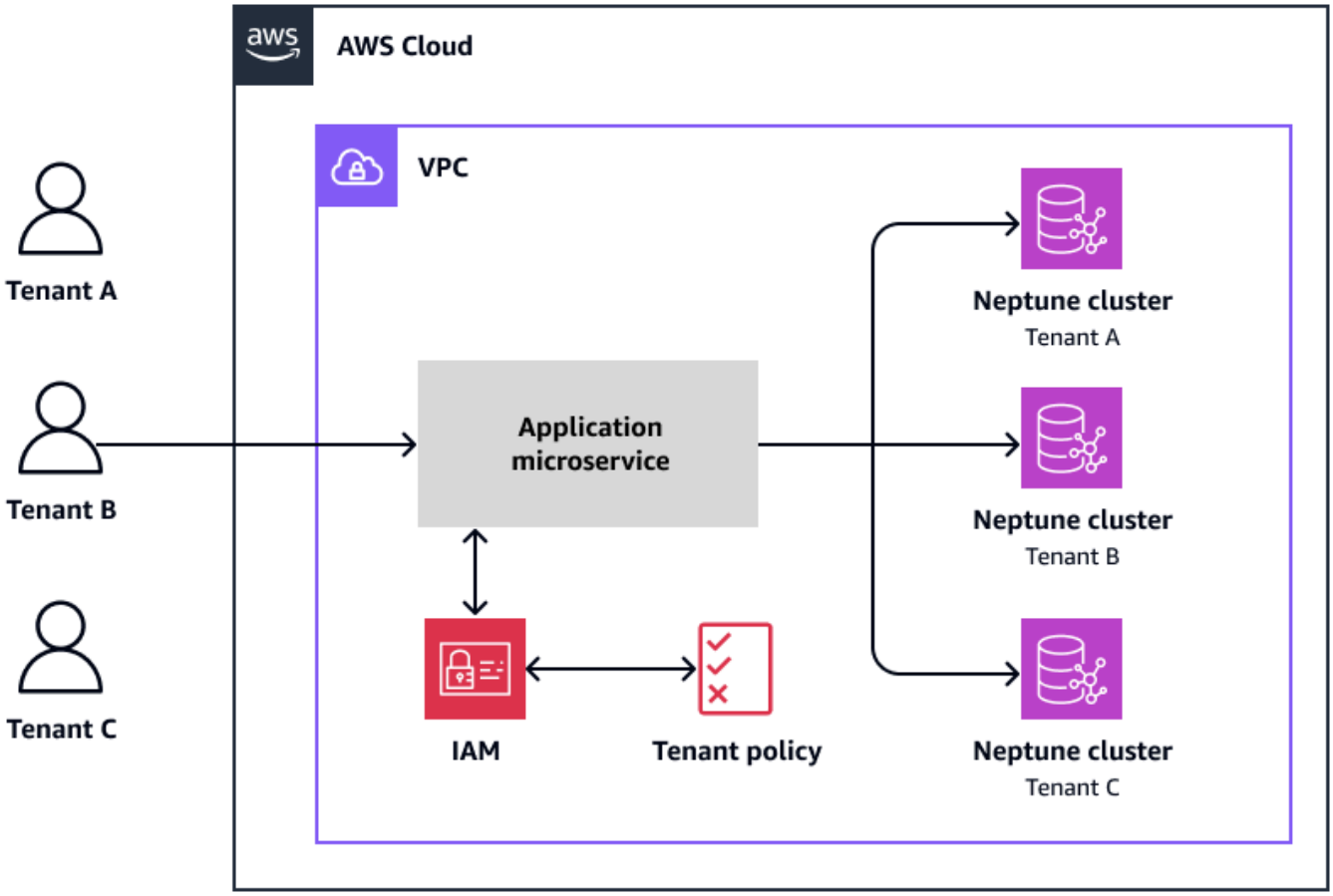
サイロモデルマルチテナンシー

一部のマルチテナント SaaS 環境では、コンプライアンスと規制要件により、テナントのデータを完全に分離されたリソースにデプロイする必要がある場合があります。大規模なお客様は、ノイズの多い近隣への影響を軽減するために専用クラスターを必要とする場合があります。このような状況では、サイロモデルを適用できます。

サイロモデルでは、テナントデータのストレージは他のテナントデータから完全に分離されます。テナントのデータを表すために使用されるすべてのコンストラクトは、そのクライアントに物理的に一意であると見なされます。つまり、各テナントには通常、個別のストレージ、モニタリング、管理があります。各テナントには、暗号化用の個別の AWS Key Management Service (AWS KMS) キーもあります。Amazon Neptune では、サイロはテナントごとに 1 つのクラスターです。

テナントあたりのクラスター

クラスターごとに 1 つのテナントを持つことで、Neptune でサイロモデルを実装できます。次の図は、仮想プライベートクラウド (VPC) 内のアプリケーションマイクロサービスにアクセスする 3 つのテナントと、テナントごとに個別のクラスターを示しています。



各クラスターには、効率的なデータインタラクションと管理のために個別のアクセスポイントを確保するのに役立つ個別のエンドポイントがあります。各テナントを独自のクラスターに配置することで、テナント間に明確に定義された境界を作成し、データが他のテナントのデータから正常に分離されるようにします。この分離は、厳格な規制とセキュリティの制約がある SaaS ソリューションにも魅力的です。さらに、各テナントに独自のクラスターがある場合、ノイズの多い近隣について心配する必要はありません。1 つのテナントが負荷を課し、他のテナントのエクスペリエンスに悪影響を及ぼす可能性があります。

cluster-per-tenantサイロモデルには利点がありますが、管理と俊敏性の課題も生じます。このモデルの分散性により、テナントアクティビティとすべてのテナントの運用状態を集約して評価することが困難になります。新しいテナントをセットアップするには別のクラスターのプロビジョニングが必要になるようになったため、デプロイも難しくなります。クライアントのアップグレードとバージョンがデータベースのアップグレードと緊密に結合されている場合、クライアントレイヤーが共有されている環境ではアップグレードがより困難になります。

Neptune は、サーバーレスクラスターとプロビジョニングされたクラスターの両方をサポートしています。アプリケーションワークロードがサーバーレスインスタンスまたはプロビジョニングされ

たインスタンスによってより適切に処理されているかどうかを評価します。一般的に、ワークロードの需要が一定である場合、プロビジョニングされたインスタンスはコスト効率が高くなります。サーバーレスは、要求の厳しい可変性の高いワークロードに最適化されており、データベースの使用量が短時間多くなると、軽いアクティビティが長時間続くか、アクティビティがない状態になります。

テナントごとに Neptune でプロビジョニングされたクラスターを使用する場合は、テナントの需要の最大負荷に近似するインスタンスサイズを選択する必要があります。このサーバーへの依存は、SaaS 環境のスケーリング効率とコストにもカスケード的に影響します。SaaS の目標は実際のテナント負荷に基づいて動的にサイズ設定することですが、Neptune プロビジョニングクラスターでは、使用量の多さと負荷の急増を考慮して過剰プロビジョニングする必要があります。オーバープロビジョニングにより、テナントあたりのコストが増加します。さらに、テナントの使用状況が時間の経過とともに変化するにつれて、クラスターのスケールアップまたはスケールダウンをテナントごとに個別に適用する必要があります。

Neptune チームは通常、アイドル状態のリソースによって発生するコストが高くなり、運用が複雑になるため、サイロモデルに対してアドバイスします。ただし、規制の厳しいワークロードや機密性の高いワークロードでは、この追加の分離が必要になるため、お客様は追加料金を支払うことができます。

サイロモデルの実装ガイダンス

テナントcluster-per-tenantサイロ分離モデルを実装するには、 AWS Identity and Access Management (IAM) データアクセスポリシーを作成します。これらのポリシーは、テナントが独自のデータを含む Neptune クラスターにのみアクセスできるようにすることで、テナントの Neptune クラスターへのアクセスを制御します。各テナントの IAM ポリシーを IAM ロールにアタッチします。次に、アプリケーションマイクロサービスは IAM ロールを使用して、 AWS Security Token Service () の `AssumeRole`メソッドを使用してきめ細かな一時的な認証情報を生成しますAWS STS。これらの認証情報は、そのテナントの Neptune クラスターにのみアクセスでき、テナントの Neptune クラスターへの接続に使用されます。

次のコードスニペットは、データベースの IAM ポリシーのサンプルを示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
```

```
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:tenant-1-cluster/*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": "arn:aws:iam::123456789012:role/tenant-role-1"
        }
      }
    }
  ]
}
```

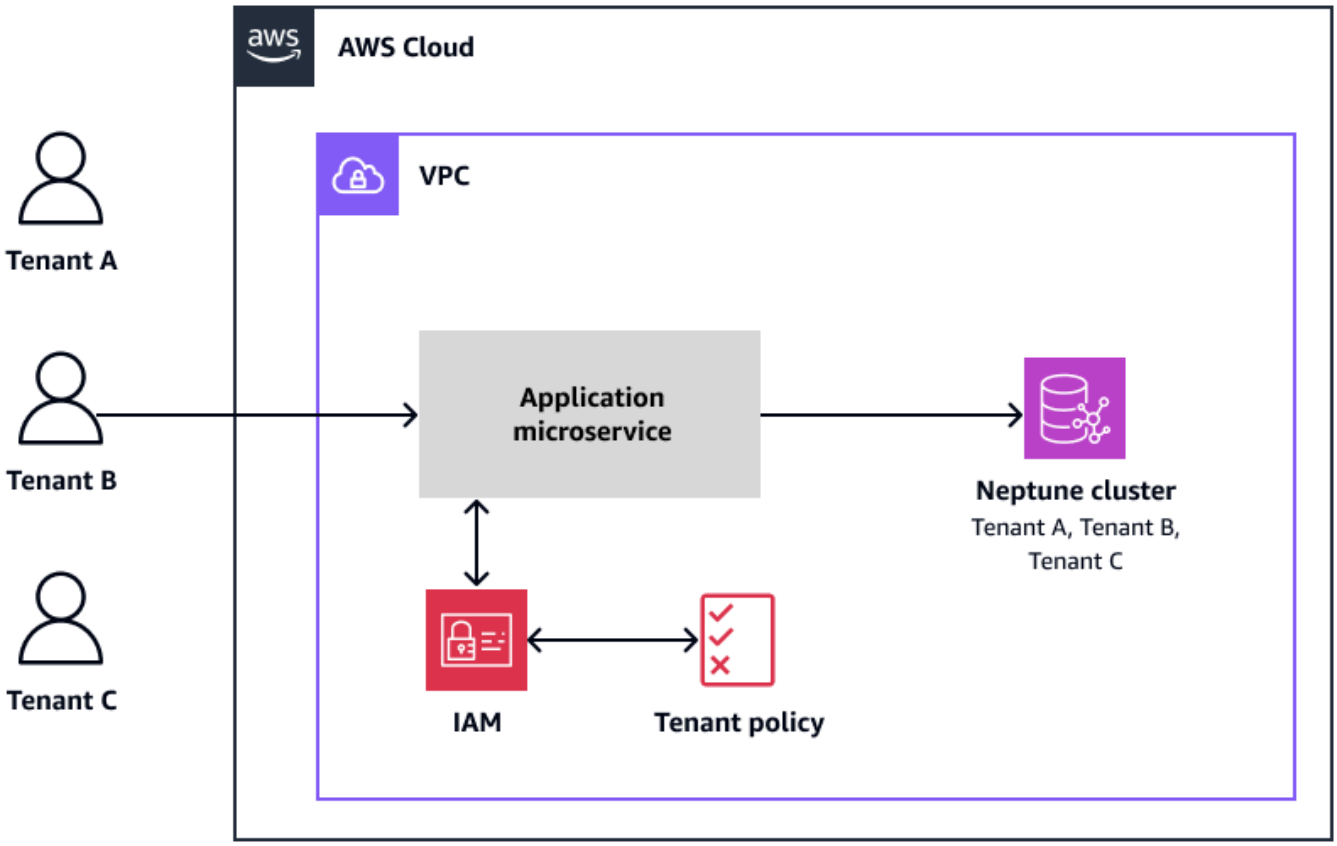
このコードは、サンプルテナントである にtenant-1、それぞれの Neptune クラスターへの読み取りおよび書き込みクエリアクセスを提供します。Condition エレメントにより、IAM ロール () tentant-1 を引き受けた呼び出し元のエンティティ (プリンシパルtenant-role-1) のみがtenant-1の Neptune クラスターにアクセスできるようになります。

プールモデルマルチテナンシー

コストや運用上のオーバーヘッドにより、サイロモデルを実装する必要や実現不可能な場合があります。

- テナントごとに個々のクラスターを維持するリソースがない可能性があります。
- 各テナントのデータを物理的に分離する必要はなく、論理的に分離するだけでニーズとコンプライアンス要件を満たすことができます。

次の図は、テナントデータが単一の Amazon Neptune クラスターに配置され、すべてのテナントが共通のデータベースを共有するプールモデルを示しています。



このプール分離モデルは、管理するクラスターが少ないため、管理オーバーヘッドを削減し、運用効率を向上させることができます。また、コンピューティングリソースは、顧客の非アクティブ期間中もアイドル状態のままではなく、複数の顧客間で共有できます。

プールモデルを使用する場合、データをモデル化する方法は 2 つあります。アプローチは、ラベル付きプロパティグラフ (LPG) を構築するか、リソース記述フレームワーク (RDF) を使用してグラフを構築するかによって異なります。

ラベル付きプロパティグラフのプールモデル

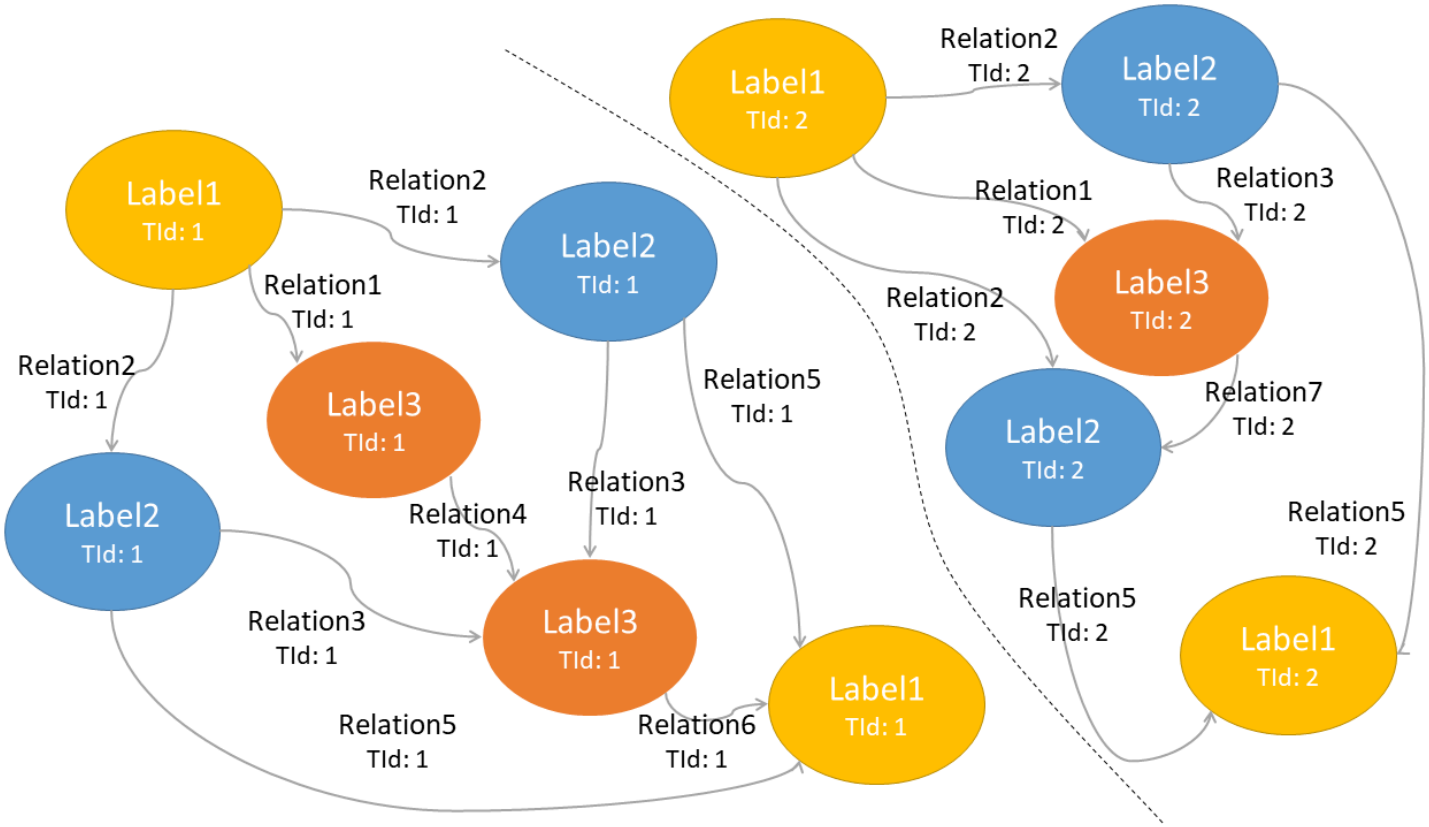
Amazon Neptune の LPGsには 3 つの異なるアプローチがあります。

- プロパティ戦略 - Apache TinkerPop Gremlin 言語の PartitionStrategy など、確立されたライブラリコンストラクトの使用をパフォーマンスよりも優先する必要がある場合は、プロパティ戦略を選択します。
- プレフィックスラベル戦略 – パフォーマンスとノイズの多い近隣効果の制限に基づいて、ほとんどのシナリオでプレフィックスラベル戦略をお勧めします。
- マルチラベル戦略 – マルチラベル戦略では、プレフィックスラベル戦略のパフォーマンスが向上しています。また、クラスター上のすべてのテナントにまたがるクエリの実行もサポートされています (たとえば、すべてのテナントでレポートまたはモニタリングするための ISV クエリ)。

プロパティ戦略

LPGs、ユーザーはキーと値のペアのプロパティをノード、頂点、エッジに追加できます。論理的な分離を実現するために、ほとんどのお客様は、これをすべてのノードと エッジで共通のテナントプロパティキーを持つ一意のプロパティとして直感的にモデル化します。テナントプロパティキーは、ノードを所有するすべてのテナントを表します。テナント識別子は、個々のテナントを識別する一意の値です。

次の図は、このモデルを示しています。切断された 2 つのサブグラフには、さまざまなラベル付きノードとエッジがあり、テナントプロパティキーは で表されますTId。1 つのサブグラフのすべてのノードとエッジのTId値は です1。もう 1 つのサブグラフでは、すべてのノードとエッジのTId値はです2。



ラベル付きプロパティグラフ内では、これを管理する方法が 2 つあります。Gremlin クエリ言語は、 データのデータパーティショニングの管理に役立つ PartitionStrategy トラバーサルライブラリを提供します。次の例のコードでは、すべてのノードとエッジに というプロパティがあることを想定していますTId。

```
strategy1 = new PartitionStrategy(partitionKey: "TId", writePartition: "1",
 readPartitions: ["1"])
strategy2 = new PartitionStrategy(partitionKey: "TId", writePartition: "2",
 readPartitions: ["2"])
```

新しいノードまたはエッジが書き込まれると、 プロパティ"TId"は、 "1" または のどちらが選択されたかに応じて"2"、 strategy1または の値で追加strategy2されます。が "TId"のお客様は"1"、 を使用しますstrategy1。次の例は、その顧客のデータの書き込みを示しています。

```
g.withStrategies(strategy1).addV("Label1").property("Value", "123456").property(id,
 "Item_1")
```

読み取りクエリの場合、 "TId == '1'" または のフィルタ"TId == '2'"はstrategy2、それぞれ strategy1または を使用してすべてのノードまたはエッジトラバーサルに追加されます。これら

のパーティション戦略はコードを簡素化しますが、必須ではありません。戦略を使用する利点は、認可レベルで挿入し、クエリを形成する下位レベルのコードに渡すことができることです。これにより、顧客識別子 (TId) を決定するコードとクエリのロジックが分離されます。

次のコード例は、データを読み取るための Gremlin クエリを示しています。

```
g.withStrategies(strategy1).V().hasLabel("Label1")
```

上記のコードは、次の例と同等です。

```
g.V().hasLabel("Label1").has("TId", "1")
```

同様に、Gremlin を使用してデータを書き込む場合は、次のクエリを使用できます。

```
g.withStrategies(strategy1).addV("Label1").property("Value").property(id, "Item_1")
```

上記のコードは、パーティション戦略を使用しないため、 "TId"プロパティを明示的に記述する必要がある次の例と同等です。

```
g.addV("Label1").property("TId", "1").property("Value").property(id, "Item_1")
```

openCypher では、これらのライブラリは存在しません。テナント識別子をノードとエッジのプロパティとして追加するには、クエリを記述して変更する必要があります。例えば、次のようになります。

```
CREATE (n:Item {`~id`: 'Item_1', Value: '123456', TId: '1'})
CREATE (n:Item {`~id`: 'Item_2', Value: '123456', TId: '2'})
```

パーティション戦略を使用しない Gremlin コード間の類似性に注意してください。その後、次のコードを使用して、最初のCREATEステートメントから書き込まれたノードを読み取ることができます。

```
MATCH (n:Item {TId: '1'})
RETURN n
--or
MATCH (n:Item)
WHERE n.TId == '1'
RETURN n
```

PartitionStrategy などのネイティブ TinkerPop Gremlin コンストラクトを使用する場合は、プロパティ戦略を選択できます。ただし、このモデルには、プレフィックスラベル戦略と比較して、Amazon Neptune のパフォーマンス上の欠点があります。これらのパフォーマンスの欠点については、「LPG モデルのパフォーマンスへの影響」セクションを参照してください。

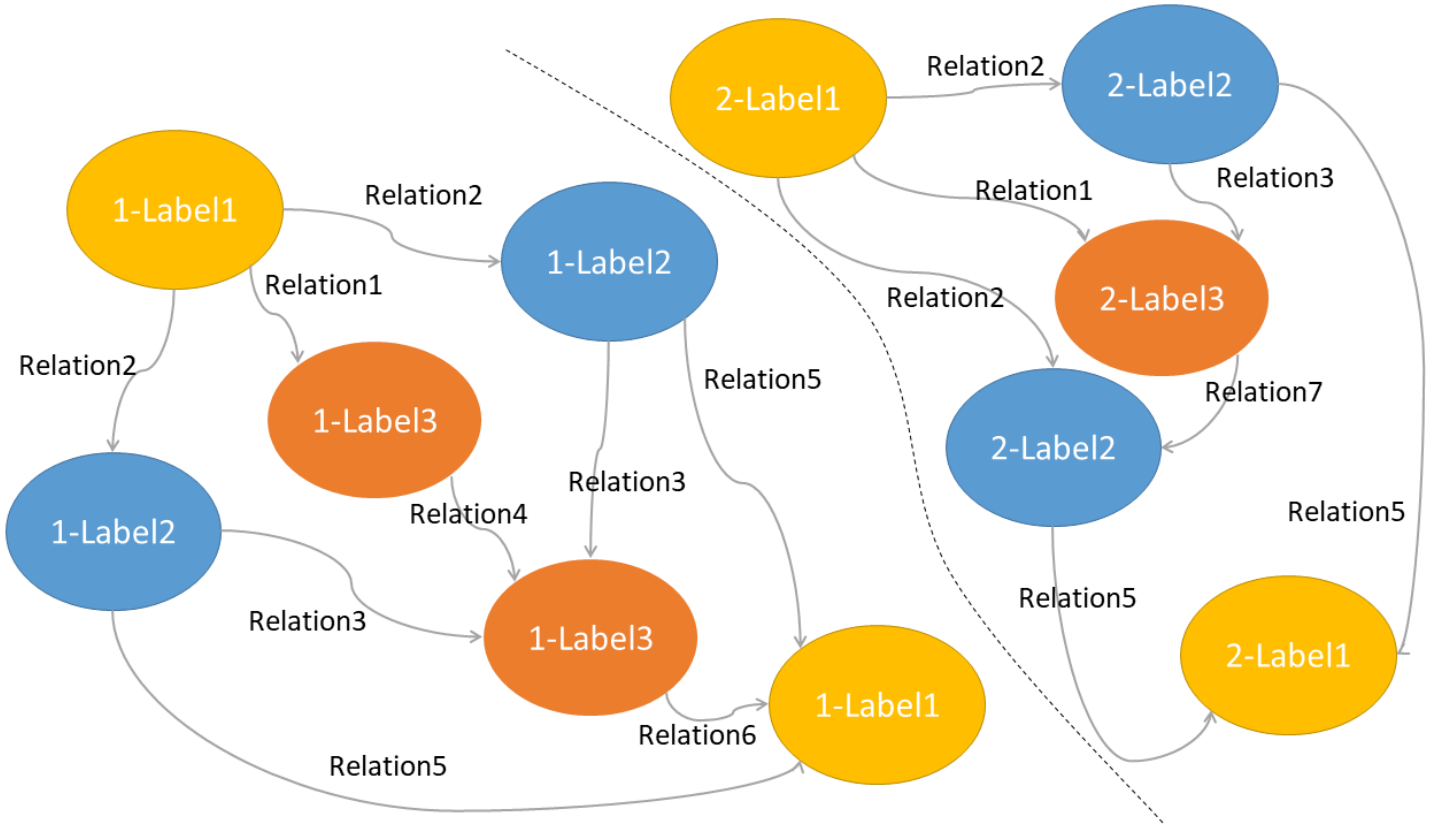
次の条件が適用される場合は、プロパティ戦略をエッジではなくノードでのみモデリングすることを検討してください。

- グラフのエッジはラベルよりも大幅に多くなります。
- 各テナントは切断されたグラフです。
- グラフにアクセスするには、ラベルではなくノードを開始点として使用します。

プレフィックスラベル戦略

パフォーマンスが最大の懸念事項である場合は、プロパティ戦略よりもプレフィックスラベル戦略を検討することを強くお勧めします。

プレフィックスラベル戦略では、各ノードにテナント識別子とノードラベルの組み合わせでラベル付けします。たとえば、テナントの識別子が "1"で、ノードラベルが の場合"Label1"、ノードラベルを として指定します"1-Label1"。次の図は、このモデルを使用する 2 つの切断されたサブグラフを示しています。



Gremlin でデータを書き込むときは、任意のノードのラベルに識別番号を追加できます。

```
g.addV("1-Label1")
g.addV("2-Label6")
```

このグラフをクエリするときは、ノードにこのプレフィックスが存在するかどうかを確認できます。

```
g.V().hasLabel("1-Label1")
```

openCypher では、 CREATEステートメントを使用してデータを書き込むことができます。

```
CREATE (n:`1-Label1` {`~id`: 'Item_1', Value: 'XYZ123456'})
```

openCypher で書き込んだデータをクエリするには、次のコードを使用します。

```
MATCH n= (:`1-Label1`)
RETURN n
```

プレフィックスラベル戦略では、すべてのノードが 1 つ以上のテナントに割り当てられ、アクセス許可がエッジスコープに割り当てられていないことを前提としています。多数の述語を引き起こ

し、Neptune のパフォーマンスに悪影響を及ぼすため、エッジラベルにこの戦略を使用しないでください。

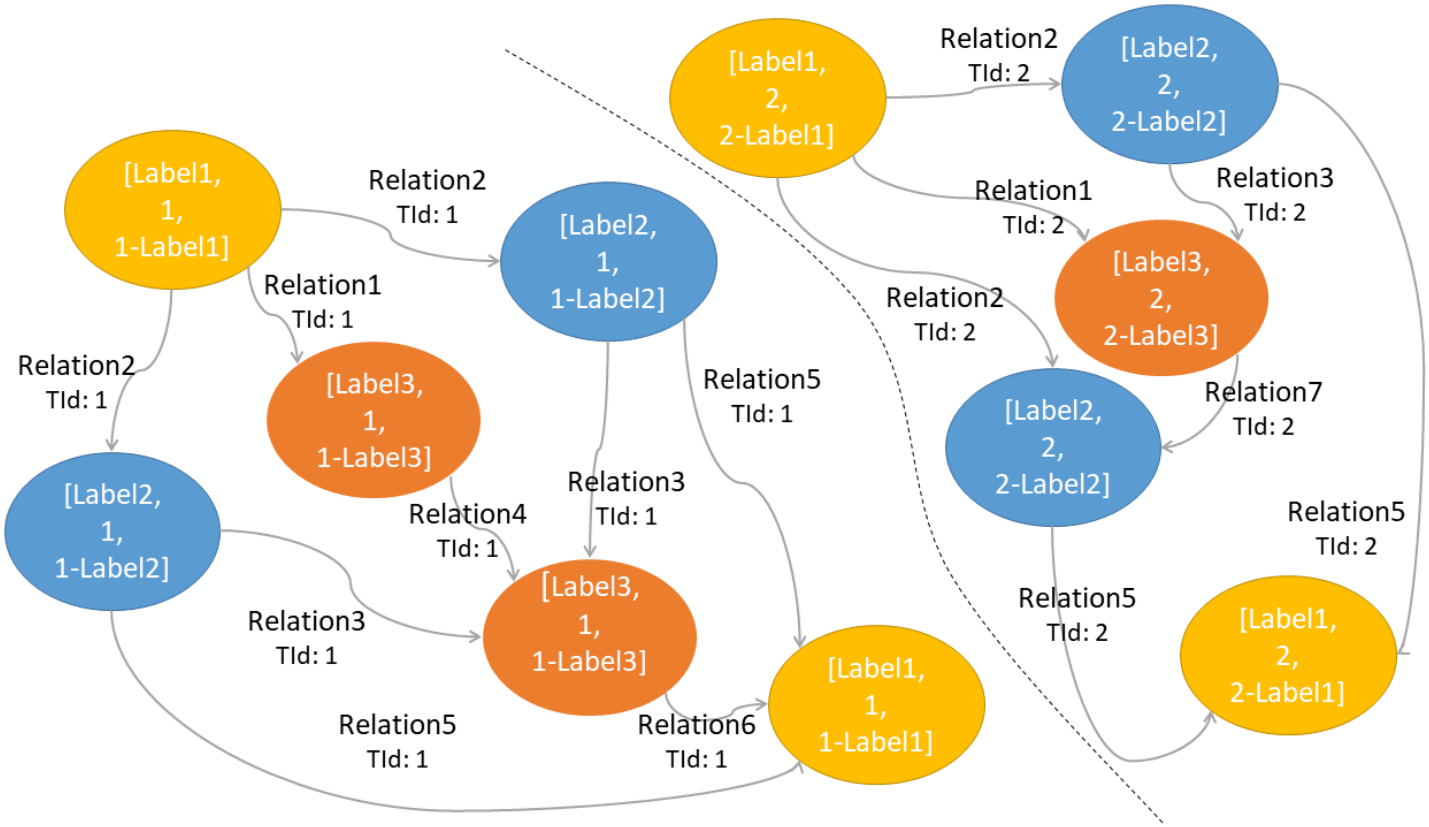
プレフィックスラベルアプローチには 2 つの主な欠点があります。まず、テナント間でクエリを実行することは困難です。一例は、レポートまたはモニタリングのために特定のラベルのすべてのノードをカウントするクエリです。これがユースケースである場合は、この戦略を複数ラベル戦略と組み合わせることを検討してください。戦略の組み合わせの詳細については、「ハイブリッドモデル」セクションを参照してください。

次に、プレフィックスラベル戦略では、データ漏洩を防ぐために、すべてのクエリに適切なプレフィックスを適切に適用するコントロールが必要です。ただし、この戦略は低レイテンシーのクエリを必要とするワークロードにとって最も効率的なオプションであり、強くお勧めします。LPG モデルのパフォーマンスへの影響セクションでは、これが最も効率的な戦略である理由の例を示します。

マルチラベル戦略

3 番目のオプションは、複数ラベル戦略を使用することです。このアプローチでは、グラフ上のすべてのノードにラベルを追加します。たとえば、特定のテナントのすべてのデータをフィルタリングする必要がある場合は、テナント ID ラベルを追加します。テナントに関係なく、特定のラベルのすべてのデータをフィルタリングする必要がある場合は、そのラベルを追加します。次の図は、ノードごとに 3 つのラベルを使用して適用される複数ラベル戦略を示しています。

次の 3 つの異なるパターンを使用してグラフにアクセスできるようになりました。



- をフィルタリングLabel1して、すべてのテナントLabel1で を持つすべてのノードを返します。
- をフィルタリング1して、テナント 1 のすべてのノードを返します。
- をフィルタリング1-Label1して、ラベルが のテナント 1 のみのすべてのノードを返しますLabel1。

LPGs、これを実装する方法は 2 つあります。

Gremlin では、SubgraphStrategy というトラバーサル戦略を使用して、すべてのクエリの範囲を などの特定のラベルを持つ頂点のみに制限できます"Label1"。

```
g.withStrategies(
    new SubgraphStrategy(
        vertices=hasLabel("Label1")
    )
)
```

PartitionStrategy とは異なり、SubgraphStrategy はデータの読み取りにのみ影響し、データの書き込みには影響しません。データを書き込むには、各クエリにラベルを手動で割り当てます。

```
g.addV("Label1").property("Value","XYZ123456")
.addV("Label2").property("Value","XYZ123456")
```

データを読み取るときは、SubgraphStrategy を使用して ですべてのノードをクエリできます"Label1"。

```
g.withStrategies(
new SubgraphStrategy(vertices=.hasLabel("Label1"))
).
V().has("Value","XYZ123456")
```

Neptune は、 "Label1"と の値を持つ最初のレコードのみを返します"XYZ123456"。これは、SubgraphStrategy を使用しない次のクエリに相当します。

```
g.V().hasLabel("Label1").hasValue("XYZ123456")
```

この基本的なクエリでは、SubgraphStrategy の使用がより複雑であるように見えます。ライブラリは、既に定義された戦略gで のインスタンスを提供できることに注意してください。開発者は、適切なフィルターが適用されていることを確認する必要はありません。

```
def getGraphTraversal():
return g.withStrategies(new SubgraphStrategy(vertices=.hasLabel("Label1"))

getGraphTraversal().has("Value","XYZ123456")
```

openCypher ライブラリにはこれらのコンストラクトがないため、ノードごとに複数のラベルを作成する必要があります。

```
CREATE (n:`1`:`Label1`:`1-Label1` {`~id`: 'Item_1', Value: '12345'})
```

これらのラベルを使用してサブグラフをフィルタリングする場合、探している顧客ラベルを持つノード、またはそのラベルを持つ別のノードとの関係を共有するノードを返すことができます。

```
MATCH n=(:Label1:`1`)
// or
MATCH n=(:`1-Label1`)
```

マルチラベル戦略を使用すると、ノードをタイプ (Label1) またはテナント () でクエリしたり、パフォーマンスが最も重要である場合により効率的なプレフィックスラベル戦略 (1) を使用したりする柔軟性が高まります1-Label1。

この戦略の主な欠点は、各ラベルがグラフに保存されている追加のオブジェクトであることです。オブジェクトは、LPGs。取り込み速度は 1 秒あたりのオブジェクトによって測定され、バインドされます。ストレージコストは消費されたギガバイト数によって異なります。つまり、追加のオブジェクトは大規模な測定可能な影響を与える可能性があります。

LPG モデルのパフォーマンスへの影響

Amazon Neptune AWS のスキルビルダーコースのデータモデリングでは、Neptune データモデルの内部とモデリングの影響について詳しく説明しますが、これらの設計に関する重要な考慮事項をここで要約します。 Amazon Neptune 1 つの Neptune クラスターに 3 つのテナント (T1、T2, T3) を持つことを検討してください。これらのテナントには次の属性があります。

- テナント 1 (T1) には合計 1 億のノードがあり、1,000 万のノードは Item 型です。
- テナント 2 (T2) には合計 1,000 万ノードがあり、100 万ノードは Item 型です。
- Tenant 3 (T3) には合計 1 億のノードがあり、100 万のノードは Item 型です。

プロパティ戦略を使用して、テナント 3 の項目を取得するクエリを実行します。Neptune は、2 つのインデックス呼び出しの統計を検査します。

- `tenant property key=T3` に 1 億件の結果がある
- `label = Item` に 1,200 万件の結果がある (T1 から T1,000 万件 + T2 から 100 万件 + T3 から 100 万件)

Neptune クエリオプティマイザは、後者のクエリが最初に適用されるのが最適であると判断し (1,200 万の結果) 、各項目で を検査しますtenant property key=T3。1,200 万件の項目を取得して、100 万件の結果を見つけます。

このクエリのノイズの多い近隣の影響に注目してください。テナントあたり 1 億のアイテムノードがある場合、最初のクエリの結果は 1,200 万ではなく 3 億になります (これは説明のために過度に単純化されています。 Neptune オプティマイザは、異なる順序のオペレーションを適用した可能性があります) 。

次に、プレフィックスラベル戦略を検討します。100 万件の結果を 返す で label=T3-Item1 回のインデックス呼び出しを行います。これにより、プロパティ戦略と同じ結果が得られますが、取得す

るレコードは 1,100 万件少なくなります。さらに、ラベルがインデックス内で重複しないため、ノイズの多い近隣の懸念がなくなりました。

マルチラベル戦略では、プロパティ戦略よりもクエリパフォーマンスが直接向上することはありません。プロパティ値によるフィルタリングは、検索スペースが比較可能な場合、ラベル値によるフィルタリングと同等です。代わりに、複数ラベル戦略はより柔軟性をサポートします。 マルチラベル戦略は、 `label=T3`または ラベル のプレフィックスラベル戦略と同等のパフォーマンスを提供しますT3-Item。マルチラベル戦略は、 のプロパティ戦略と同等のパフォーマンスを提供しますlabel=Item。利点は、さまざまなアクセスパターンをサポートすることです。

RDF のプールモデル

Resource Description Framework (RDF) には、名前付きグラフの概念があり、データを論理的に分離する方法を提供します。Amazon Neptune には、デフォルトの名前付きグラフとユーザー定義の名前付きグラフがあります。名前付きグラフは必要な数だけ作成できます。総称して、RDF データセットと呼ばれます。デフォルトまたはユーザー定義のすべての名前付きグラフは、RDF データセット内の国際化リソース識別子 (IRI) によって定義されます。Neptune では、ユーザーがデータの書き込み時に名前付きグラフを宣言しない限り、すべてのトリプルはデフォルトの名前付きグラフの一部と見なされます。

名前付きグラフには複数のユースケースがあります。

- データパーティショニングとデータ分離
- データ出典
- バージョニング
- 推論

このガイドでは、データパーティショニングのユースケースに焦点を当てています。テナントごとに 1 つのユーザー定義の名前付きグラフを作成することをお勧めします。

Graph Store HTTP Protocol を使用した SPARQL クエリオプション

次のクエリ例では、SPARQL プロトコルと RDF クエリ言語 (SPARQL) と Graph Store HTTP プロトコルを使用して、テナントの名前付きグラフをクエリまたは作成します。

- `HTTP GET` – テナントの特定のグラフを取得するには:

```
curl --request GET 'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//
www.example.com/named/tenant1'
```

- HTTP PUT – 特定の名前付きグラフを作成またはリクエストで指定されたペイロードに置き換えるには:

```
curl --request PUT -H "Content-Type: text/turtle" \  --data-raw "@prefix ex: http://
example.com/ . ex:subject ex:predicate ex:object ." \
'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

  RDF では、オブジェクトはトリプルです。

- HTTP POST – 名前付きグラフが存在しない場合は新しいグラフを作成するか、既存のグラフとマージするには:

```
curl --request POST -H "Content-Type: text/turtle" \
--data-raw "@prefix ex: http://example.com/ . ex:subject ex:predicate ex:object ." \

'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

RDF のテナント分離

アプリケーションレイヤーに必要なガードレールがあるデータを論理的に分離するには、テナントとユーザー定義の名前付きグラフ間のマッピングを作成します。RDF データセットのマルチテナンシーを設計する場合は、RDF と SPARQL の次の側面に注意してください。

- Neptune では、名前付きグラフを指定せずにクエリを実行すると、データベース内のすべての名前付きグラフでパターンに一致するすべてのトリプルを取得します。
- RDF では、異なる名前付きグラフのノード間の接続に制約はありません。例えば、前の図では、のノードを エッジを介して G2のノードに接続:G1できます。

たとえば、特定のテナントのエンドユーザーが API にクエリを送信する場合、API は Neptune データベースにクエリを送信する前に次の要件を検証する必要があります。

- 単一のテナントを対象とするクエリでは、名前付きグラフを指定する必要があります。そうしないと、テナント間でデータが漏洩するリスクがあります。

- 更新または削除クエリでは、常に名前付きグラフを指定する必要があります。
- エッジまたはリレーションシップの両側にあるノードは、常に正しい名前付きグラフに属している必要があります。

ベストプラクティスの詳細については、Neptune ドキュメントを参照してください。

成長に備える

プールモデルを正常に使用すると、最終的に単一の Neptune クラスターのサイズが大きくなります。テナントが増加するか、テナントの数が増加し、すべての顧客に必要なデータの取り込み率がクラスターの機能を超えています。これが発生した場合は、顧客を複数のクラスターに分割する必要があります。後で改良を試みるのではなく、この設定を事前に設計してください。初期スケールが 1 つのクラスターのみを使用する場合でも、そのスケールに達したときに将来複数のクラスターにテナントをルーティングする必要があるコンポーネントをモックアップします。

テナントのサイズに基づいてソリューションにより多くのリソースが必要な場合は、その成長に備えてください。1 つのクラスターの複数のお客様が大幅に増加した場合、そのクラスターは要件をサポートしなくなる可能性があります。Amazon Neptune DB クローン作成機能を使用して、テナントを別のクラスターに移動するか、既存のクラスターを 2 つに分割する戦略を設計します。

DB クローンを実装するときにコストを削減できる Neptune Copy-on-Write Protocol に精通してください。取り込みのボトルネックのためにクラスターを分割すると、ポリシーで許可されている限り、クラスターからデータを削除しない方が効率的です。2 つのクラスターは、データページが変更されていない場合はデータページを共有しますが、データページが変更されている場合は共有しません (データページの一部が削除されたため)。

> **Note**
>
> このガイダンスは、この執筆時の最新の Neptune バージョンである Neptune バージョン 1.3.1 に適用されます。このガイダンスは、Neptune ストレージレイヤーが進化するにつれて、将来のバージョンで変更される可能性があります。

マルチテナンシーシナリオの制限事項

一部の Neptune 機能は、マルチテナンシーシナリオ用に構築されていないことに注意してください。これらのマルチテナンシー戦略はデータベースレベルで強制されないため、テナントにはプール

モデル内の Neptune エンドポイントへの直接アクセスを許可しないでください。このドキュメントで説明されている設計を適用する Neptune エンドポイントと顧客の間には、常に何らかのプロキシを保持してください。このようなプロキシの例は次のとおりです。

- クライアントレイヤーにラベルフィルターを追加する
- 認証トークンをテナント ID にマッピングし、このフィルターをクエリに挿入する API がある

このガイダンスは、Neptune グラフノートブック、Neptune グラフエクスプローラー、Neptune ストリームなどの機能に直接アクセスできるようにする場合にも適用されます。

ハイブリッドモデルマルチテナンシー

SaaS ソリューションは、多くの場合、サイロモデルとプールモデルの組み合わせを使用します。さまざまな要因が、同じ環境内でサイロモデルとプールモデルの両方をいつどのように採用するかの決定に影響を与えます。

このような要因の 1 つは階層化です。SaaS SaaS ソリューションは、テナントの各階層に独自のエクスペリエンスを提供します。例えば、階層が無料、スタンダード、プレミアムの場合、無料階層のテナントデータはプールモデルを使用して共有 Neptune クラスターに保存できます。スタンダードおよびプレミアム階層テナントでは、 cluster-per-tenant サイロモデルを使用できます。

さらに、一部の SaaS プロバイダーには、基盤として共有 Amazon Neptune クラスターにプールソリューションを構築できる機能があります。その後、多くの場合、コンプライアンスと規制の義務により、サイロ化されたストレージを必要とするテナント用に個別の Neptune クラスターを作成できます。

これにより、データアクセスレイヤーと管理プロファイルにある程度の複雑さが加わる可能性がありますが、お客様の要件を満たすために、提供サービスを階層化する方法をビジネスに提供することもできます。

ISVsの運用上のベストプラクティス

このセクションのガイドラインの多くは、すべてのお客様向けのベストプラクティスですが、ISVsには重要性が付加されています。

Neptune クラスターを最新バージョンで更新する

Amazon Neptune リリースノートでは、すべてのバージョンが多数のバグ修正、パフォーマンスの向上、新機能をもたらしていることがわかります。Neptune クラスターをできるだけ最新バージョンに保ちます。

以前に検出されなかったバグがワークロードにあり、クラスターが最新バージョンである場合、Neptune エンジニアはクラスターのプライベートパッチを作成できます (必要に応じて) 。パッチは、その修正が一般公開される次のリリースまでブリッジできます。クラスターを最新バージョンに更新するには、Neptune Blue/Green ソリューションを使用します。

データインジェストに削除と置換の代わりに差分を使用する

Neptune にデータを取り込んだり書き込んだりするには、いくつかの手法を使用できます。多くのお客様は、フィードで変更が受信されるたびにグラフを削除して再挿入することで、データの取り込みを簡素化しようとしています。各ノードに`last-modified`プロパティを追加し、指定した日付以降に変更されていないノードを定期的にスキャンして削除する場合があります。これらの手法はデータ取り込みプロセスを簡素化しますが、Neptune クラスターには長期的なヘルスとスケーラビリティの影響があります。

まず、Neptune は文字列のディクショナリエンコーディングを使用します。ノードとエッジの IDs を明示的に指定しない限り、Neptune は ID の文字列として表される GUID を生成し、その文字列をディクショナリに保存します。オブジェクトを常に削除して追加している場合、自動的に生成された IDs によってディクショナリが肥大化します。

次に、Neptune は最大 1 秒あたり約 120 K オブジェクトを取り込むようにスケールアップします。オブジェクトを継続的に削除して追加する場合、基本的に変更されていないオブジェクトでその帯域幅の多くを消費します。これにより、クラスターでホストできるテナントの数が制限され、クラスター内により大きなライターインスタンスが必要になり、より多くの I/O オペレーションが必要になります。これらの要因はすべてコストを増加させます。

削除および追加メソッドを使用する代わりに、変更内容の真の差分を計算する方法を開発することを強くお勧めします。ただし、一部のデータソースはこれには適していません (たとえば、現在の状

態を返す API コールや、変更された内容を正確に追跡しないイベントなど) 。raw データソースが変更の特定に適していない場合は、抽出、変換、ロード (ETL) プロセスを使用して差分を計算します。例えば、以前の各データキャプチャのスナップショットを Parquet 形式で維持 AWS Glue し、を使用してそれらのスナップショットの違いを計算し、その違いのみを Neptune にプッシュできます。

Neptune のコストがテナントとともにどのように進化するかをモデル化する

サイロモデル、プールモデル、ハイブリッドモデルのいずれを使用する場合でも、クラウドコストはテナントのサイズに応じてスケールします。同時接続数が多いテナントでは、同時接続数が少ないテナントよりも大きなインスタンスまたはリードレプリカが必要です。より迅速なデータ取り込みを必要とするテナントにも同じことが当てはまります。

Neptune クラスターコストの 3 つのコンポーネントは、インスタンスサイズ (および数) 、データサイズ (GB/月) 、I/O オペレーション (100 万あたり) です。これらのコストは一般的にワークロード固有ですが、サイズとデータ量に応じてスケールしますが、 AWS ツールを使用して測定できます。時間の経過とともにサイズがどのように変化するかなど、テナントのサイズの主要な指標に照らしてスケールの経済を追跡し、理解します。I/O 料金の予測不能性がマージンに影響する場合は、より予測可能なコストで Neptune I/O 最適化ストレージを選択することを検討してください。

お客様の需要に合わせてクラスターをスケールする

Neptune インスタンスサイズを適切にサイズ設定するための試行式や true 式はありません。Neptune ドキュメントにはガイダンスが記載されていますが、変数が多すぎて直接マッピングを推奨できません。これらの変数には以下が含まれますが、これらに限定されません。

- データモデル
- データシェイプ
- クエリの同時実行数
- クエリの複雑さ。

ワークロードとテナントプロファイルに最適なサイズを決定するためのテストを計画します。一般的に、コスト効率と予測可能性のために、プロビジョニングされたインスタンスを使用することをお勧めします。カスタマーエクスペリエンスの目標がコストよりも最適なスケーリングを優先する場合

は、Neptune Serverless インスタンスを使用して、ワークロードの変動に関係なく、より一貫したエクスペリエンスを確保することを検討してください。

テナントの読み取りワークロードのピークとトラフに大きなばらつきがある場合は、Neptune Serverless インスタンスを Neptune 自動スケーリングと組み合わせてください。 通常、新しいリードレプリカが初期化されてからオンラインになるまでに 10～15 分かかります。つまり、自動スケーリングだけではトラフィックの長期的な変化を処理できますが、アクティビティの急激な急増には不十分です。Neptune サーバーレスと Neptune 自動スケーリングを組み合わせることで、インスタンスをスケールアップまたはスケールダウンしたり、リードレプリカの数をスケールインおよびスケールアウトしたりできます。

テナントのワークロードプロファイルまたはサービスレベルアグリーメント (SLAs) が大幅に異なる場合は、カスタムエンドポイントと専用リードレプリカを使用して、そのトラフィックに最適化されたインスタンスにトラフィックを誘導することを検討してください。最適化には、インスタンスの異なるサイズ設定、特定のクエリパターン、バッファキャッシュの事前ウォーミングなどがあります。

次のステップ

マルチテナンシーISVアプリケーションに Amazon Neptune を実装するジャーニーを開始したばかりの場合は、必要なモデルについてさらに考慮してください。モデルを変更すると、ジャーニーの後半でコストが高くなります。

ジャーニーの早い段階では、ニーズに最適なモデルを使用し、そのモデルのガイダンスに従っていることを確認します。

常に余裕を持って計画するようにしてください。ジャーニーの早い段階では、頂点や エッジを削除して再追加するのではなく、クラスター間での顧客のシャーディングやETLプロセスの最適化作業を延期して変更のデルタを提供するのが魅力的です。スケーリングすると、これらの決定がパフォーマンスとコストに悪影響を及ぼす可能性があります。

最後に、すでにジャーニーに慣れている場合、このガイダンスにより、アーキテクチャが最適であることが再確認されたり、アーキテクチャを改善するための変更が加えられたりする可能性があります。

このガイダンスについて質問がある場合や、さらにサポートが必要な場合は、 AWS アカウントチームに連絡して、Neptune スペシャリストとのセッションを依頼してください。

リソース

- Amazon Neptune ドキュメント
- Amazon Neptune のデータモデリング (コース)
- Amazon Neptune 用の AWS Well-Architected フレームワークの適用
- SaaS レンズ Well-Architected フレームワーク
- でのマルチテナントアーキテクチャのガイダンス AWS
- SaaS テナント分離戦略: マルチテナント環境でのリソースの分離
- Apache TinkerPop ドキュメント
- SPARQL

寄稿者

このガイドの寄稿者には以下が含まれます。

- Brian O'Keefe、プリンシパル WW SSA Neptune、 AWS
- Veeresham Gande、シニアテクニカルアカウントマネージャー、 AWS
- Dana Owens、スタートアップソリューションアーキテクト、 AWS
- Nima Seifi、スタートアップソリューションアーキテクト、 AWS

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
初版発行	—	2024 年 9 月 3 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範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する。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例： Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。
- 廃止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。アクティブ/パッシブ移行よりも柔軟な方法ですが、さらに多くの作業が必要となります。

アクティブ/パッシブ移行

ソースデータベースとターゲットデータベースを同期させながら、データがターゲットデータベースにレプリケートされている間、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。移行中、ターゲットデータベースはトランザクションを受け付けません。

集計関数

複数行に処理を行い、グループ全体を対象に単一の戻り値を計算する SQL 関数。集計関数の例としては、SUM や MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーション制御

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の重要な要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」をご覧ください。

AI オペレーション (AIOps)

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

非対称暗号化

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

原子性、一貫性、分離性、耐久性 (ACID)

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

属性ベースのアクセス制御 (ABAC)

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

信頼できるデータソース

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

アベイラビリティーゾーン (AZ)

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

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスを整理しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションのガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

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

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントの「動作グラフのデータ」を参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

それぞれが独立しているが、同一の環境を 2 つ作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (ブルー) で実行し、新しいアプリケーションバージョンを別の環境 (グリーン) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染しており、ボットハーダーまたはボットオペレーターと呼ばれる単一の当事者によって制御されているボットのネットワーク。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発した

り、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 AWS Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータを参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「ビジネス機能を中心に組織化」セクションを参照してください。

ビジネス継続性計画 (BCP)

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

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

エンドユーザーへのバージョンリリースを、時間をかけて段階的に行うこと。確信が持てたら新規バージョンをデプロイして、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

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

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストすること。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

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

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、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 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的に、エッジコンピューティングに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」を参照してください。

導入のクラウドステージ

組織が、 AWS クラウドへの移行時に通常実行する 4 つの段階。

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略との関連性については、 AWS 「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、GitHub や Bitbucket Cloud があります。コードの各バージョンはブランチと呼ばれます。マイクロサービスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

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

コールドデータ

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

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオといった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

ワークロードにおいて、設定が想定した状態から変化すること。これによって、ワークロードが非準拠になる可能性がありますが、この状態は、徐々に生じ、意図的なものではありません。

構成管理データベース (CMDB)

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

コンフォーマンスパック

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

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

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

データドリフト

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

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行えるアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼できる ID のみが、期待されるネットワークから信頼できるリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

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

データ出所

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

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには、一般的に、大量の履歴データが含まれており、多くの場合、それらはクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせます。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

深層学習

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

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

Organizations ドキュメントの「AWS Organizationsで使用できるサービス」を参照してください。

トラブルシューティング

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、「 AWSでのセキュリティコントロールの実装」の「検出的コントロール」を参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

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

ディメンションテーブル

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、クエリの制約、フィルタリング、結果セットのラベル付けに使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

ディザスタによるダウンタイムとデータ損失を最小限に抑えるための戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 Disaster Recovery of Workloads on AWS: Recovery in the Cloud」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

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

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を改善できます。

電子データ交換 (EDI)

組織間で行う、ビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティング処理。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

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

エンタープライズリソースプランニング (ERP)

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

エンベロープ暗号化

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

環境

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

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

エピック

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

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

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

F

ファクトテーブル

スタースキーマの中央にあるテーブル。ビジネスオペレーションに関する定量的データが保存されます。一般的に、ファクトテーブルは、2 種類の列で構成されます。1 つは測定値が含まれる列、もう 1 つはディメンションテーブルへの外部キーが含まれる列です。

フェイルファスト

開発ライフサイクルを短縮するために、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

では AWS クラウド、アベイラビリティーゾーン AWS リージョン、コントロールプレーン、データプレーンなどの境界で、障害の影響を制限し、ワークロードの耐障害性を向上させるのに役立ちます。詳細については、「AWS 障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

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

特徴量重要度

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

機能変換

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

数ショットプロンプト

LLM に、タスクと望ましい出力を示す例を少数提示した後に、類似のタスクを実行させること。この手法は、プロンプトに記述された例 (ショット) からモデルが学習する「インコンテキスト学

習」の一種です。数ショットプロンプトは、特定のフォーマット、推論、専門知識が必要なタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセス制御」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

大規模な深層学習ニューラルネットワークであり、一般化およびラベル付けされていないデータからなる大規模データセットでトレーニングされています。FM により、言語理解、テキストおよび画像生成、自然言語での会話といった、一般的な各種タスクを実行できます。詳細については、「基盤モデルとは何ですか?」を参照してください。

G

生成 AI

AI モデルのサブセット。大量のデータでトレーニングされており、シンプルなテキストプロンプトを使用して、画像、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できます。詳細については、「生成 AI とは何ですか?」を参照してください。

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リスト

を使って指定します。詳細については、CloudFront ドキュメントの「コンテンツの地理的ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、 AWS Security Hub CSPM、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。

高可用性 (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニング用データセットから保留される、ラベル付き履歴データの一部。ホールドアウトデータを使用すると、モデル予測をホールドアウトデータと比較して、モデルのパフォーマンスを評価できます。

同種データベースの移行

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

ホットデータ

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

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

アイドル状態のアプリケーション

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

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、ミュータブルインフラストラクチャよりも一貫性、信頼性、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

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

インダストリー 4.0

2016 年に Klaus Schwab 氏が提唱した用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩による、ビジネスプロセスのモダナイズを意味します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

インダストリアル IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」を参照してください。

インスペクション VPC

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

IoT

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

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

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

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

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

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。

大規模言語モデル (LLM)

大量のデータで事前トレーニングされた深層学習 AI モデル。LLM では、質問への回答、ドキュメントの要約、他言語へのテキスト翻訳、文を完成させるなど、さまざまなタスクを実行できます。詳細については、「大規模言語モデル (LLM) とは何ですか?」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの「最小特権アクセス許可を適用する」を参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティやプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスを招く可能性があります。マルウェアの例には、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス 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 を運用し、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。マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

「Migration Acceleration Program」を参照してください。

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。詳細については、AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

発行/サブスクライブのパターンに基づく、軽量のマシンツーマシン (M2M) 通信プロトコルであり、リソースに限りのある IoT デバイスに使用されます。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス

機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

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

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

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

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

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

移行戦略

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

ML

「機械学習」を参照してください。

モダナイゼーション

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

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、「モノリスをマイクロサービスに分解する」を参照してください。

MPA

「Migration Portfolio Assessment」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルなインフラストラクチャ

本番ワークロードに使用する既存のインフラストラクチャを更新および変更するためのモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

Open Process Communications - Unified Architecture (OPC-UA)

産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。OPC-UA により、相互運用の際に、データ暗号化、認証、認可の各スキームを標準化できます。

オペレーショナルレベルアグリーメント (OLA)

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

運用準備状況レビュー (ORR)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイドを参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail を ログに記録する によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの「組織の証跡の作成」を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイドを参照してください。

オリジンアクセス制御 (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OAC も併せて参照してください。OAC では、より詳細な、強化されたアクセス制御が可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

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

P

アクセス許可の境界

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

個人を特定できる情報 (PII)

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

PII

「個人を特定できる情報」を参照してください。

プレイブック

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

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

次の操作を可能にするオブジェクト: アクセス許可を定義する (ID ベースのポリシーを参照)。アクセス条件を指定する (リソースベースのポリシーを参照)。 AWS Organizations の組織における全アカウントにアクセス許可の上限を定義する (サービスコントロールポリシーを参照)。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行の準備状況の評価」を参照してください。

述語

`true` または `false` を返すためのクエリ条件。一般的に、`WHERE` 句に記述されます。

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、「AWSでのセキュリティコントロールの実装」の「予防的コントロール」を参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントの「ロールに関する用語と概念」にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

プライベートホストゾーン

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

プロアクティブコントロール

非準拠リソースのデプロイ防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

製品の設計、開発、発売から、成長、成熟、衰退、廃棄に至る、製品のライフサイクル全体を通してデータとプロセスを管理すること。

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

製造分野で使用される、信頼性と適応性に優れたコンピュータであり、これによって、マシンをモニタリングするとともに、製造プロセスを自動化します。

プロンプトチェイニング

1 つの LLM プロンプトによる出力を次のプロンプトの入力に使用して、より良いレスポンスを生成します。この手法を使用すると、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改良または拡張したりできます。これによって、モデルのレスポンスの精度と関連性が向上し、粒度の高いパーソナライズされた結果を得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

発行/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、マイクロサービスは、他のマイクロサービスがサブスクライブ可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

手順などの一連のステップであり、SQL リレーショナルデータベースシステムのデータにアクセスするために使用されます。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RAG

「検索拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

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

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の から分離され、独立しています。詳細については、「アカウントが使用できる AWS リージョン を指定する」を参照してください。

リグレッション

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートが含まれる場合は RASCI マトリックスと呼ばれ、含まれない場合は RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」を参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、トレーニングデータソースの外部にある信頼できるデータソースが参照されます。例えば、RAG モデルによって、組織のナレッジベースまたはカスタムデータのセマンティック検索を実行できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

定期的にシークレット情報を更新して、攻撃者が認証情報にアクセスするのをより困難にするプロセス。

行と列のアクセス制御 (RCAC)

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

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

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

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 AWS マネジメントコンソール にログインしたり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントの「SAML 2.0 ベースのフェデレーションについて」を参照してください。

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存するパスワードやユーザー認証情報などの AWS Secrets Manager機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値には、バイナリ、1 つの文字列、複数の文字列を指定できます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの概要」を参照してください。

セキュリティバイデザイン

開発プロセス全体を通してセキュリティが考慮されているシステムエンジニアリングのアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

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

セキュリティレスポンスの自動化

セキュリティイベントへの自動レスポンスまたは自動修復を目的として、事前定義およびプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例には、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る 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、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

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

責任共有モデル

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

SIEM

「Security Information and Event Management システム」を参照してください。

単一障害点 (SPOF)

特定のアプリケーションを構成する単一の重要なコンポーネントで発生し、システム稼働に支障をきたす可能性のある障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

データベースの編成構造を意味し、1 つの大きいファクトテーブルにトランザクションデータまたは測定データが保存され、1 つ以上の小さいディメンションテーブルにデータ属性が保存されます。この構造は、データウェアハウスやビジネスインテリジェンスを用途とするように設計されています。

strangler fig パターン

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

サブネット

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

監視制御とデータ取得 (SCADA)

製造分野において、ハードウェアとソフトウェアを使用して物理アセットと本番運用をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に動作を指示する手法。システムプロンプトは、コンテキストを設定して、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

トランジットゲートウェイ

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

トランクベースのワークフロー

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

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザを分け合えることができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。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

現在のレコードに何らかの形で関連している行のグループに計算を実行する SQL 関数。ウィンドウ関数は、移動平均を計算したり、現在の行の相対位置に基づいて他の行の値にアクセスするといったタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

データを 1 回のみ書き込むことで、データの削除や変更を防ぐストレージモデル。承認済みユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャは、イミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

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