



で ESG データ用のスケーラブルなウェブクローリングシステムを構築する AWS

AWS 規範ガイド



AWS 規範ガイド: で ESG データ用のスケーラブルなウェブクロー ルシステムを構築する AWS

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

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

Table of Contents

序章	1
対象者	2
ターゲットを絞ったビジネス成果	2
アーキテクチャ	3
ウェブクローラーの設計と運用	4
バッチ処理とデータ処理	6
システムの構築	7
データセットの準備	7
ウェブクローラーの構築	9
robots.txt ファイルのキャプチャと処理	9
サイトマップのキャプチャと処理	11
クローラーの設計	13
AWS インフラストラクチャの構築	20
ベストプラクティス	22
Robots.txt コンプライアンス	22
クロールレートの制限	23
ユーザーエージェントの透明性	23
効率的なクローリング	23
アダプティブアプローチ	23
エラー処理	23
バッチでのクローリング	23
セキュリティ	24
その他の考慮事項	24
よくある質問	25
robots.txt ファイルが使用できない場合はどうなりますか?	25
sitemaps.xml ファイルが使用できない場合はどうなりますか?	25
Amazon EC2 または Amazon ECS の代わりにサーバーレスソリューションを使用できますか?	26
クローラーが 403 ステータスコードを取得するのはなぜですか?	26
次のステップとリソース	27
リソース	27
ツール	27
ドキュメント履歴	28
用語集	29

#	29
A	30
B	32
C	34
D	37
E	41
F	44
G	45
H	46
I	48
L	50
M	51
O	55
P	58
Q	61
R	61
S	64
T	68
U	69
V	70
W	70
Z	71
	lxxii

で ESG データ用のスケーラブルなウェブクローリングシステムを構築する AWS

Vijit Vashishtha と Mansi Doshi, Amazon Web Services

2025 年 1 月 ([ドキュメント履歴](#))

環境、社会、ガバナンス (ESG) の要因は、潜在的な投資を評価する際に、投資家にとって重要な考慮事項です。

- 環境 – 自然界に対する企業の影響に焦点を当てます。これには、炭素排出量、リソース管理、エネルギー効率などの要因が含まれます。
- ソーシャル – 企業が従業員、サプライヤー、顧客、コミュニティとの関係をどのように管理しているかを調べます。ここでは、労働慣行、多様性、コミュニティエンゲージメントなどの側面について説明します。
- ガバナンス – 会社のリーダーシップ、内部統制、および株主の権利に注目します。これには、取締役会の構成、経営陣の補償、ビジネス活動が含まれます。

堅牢な ESG プラクティスを実践している企業は、長期的な持続可能性と収益性を重視する姿勢が強まっています。ESG 情報に対する投資家の需要が高まっています。信頼性が高く有用な ESG データを通じて持続可能性の認証情報を実証できる企業は、資本を惹きつけ、競争力を維持するのに適しています。企業は、ニュース、記事、年次レポートなど、さまざまなソースを通じて ESG データを発行します。この情報は分散されているため、ウェブクローラーはこのデータを効率的に収集するのに役立ちます。

この包括的なガイドでは [AWS Fargate](#)、[Amazon Elastic Compute Cloud \(Amazon EC2\)](#)、[AWS Batch](#)、[Amazon Simple Storage Service \(Amazon S3\)](#) を使用して、堅牢でスケーラブルで責任あるデータ収集パイプラインを構築する方法を示します。ここでは、以下について説明します。

- 次の を使用して、スケーラブルなクローリングシステムを設計します AWS のサービス。
 - クローラアプリケーションを実行するための Fargate または Amazon EC2
 - AWS Batch 大規模なクローリングジョブを効率的にオーケストレーションする
 - 安全で耐久性のあるデータストレージのための Amazon S3
- 以下を含む、倫理的クローリングのベストプラクティスを実装します。
 - robots.txt とウェブサイトポリシーの尊重

- ターゲットサイトが圧倒されないようにレート制限を管理する
- 収集された情報のデータのプライバシーと責任ある使用を確保する
- インフラストラクチャに AWS 最適化された Python ベースのクローラの開発
- 倫理基準を維持しながらクローラのパフォーマンスを最適化する

対象者

このガイドは、公開ウェブサイトから大量の up-to-date ESG データを効率的に収集したいデータエンジニアやクラウドアーキテクトを対象としています。これは、市場分析、持続可能な財務評価、または財務調査を含むプロジェクトに特に関連します。

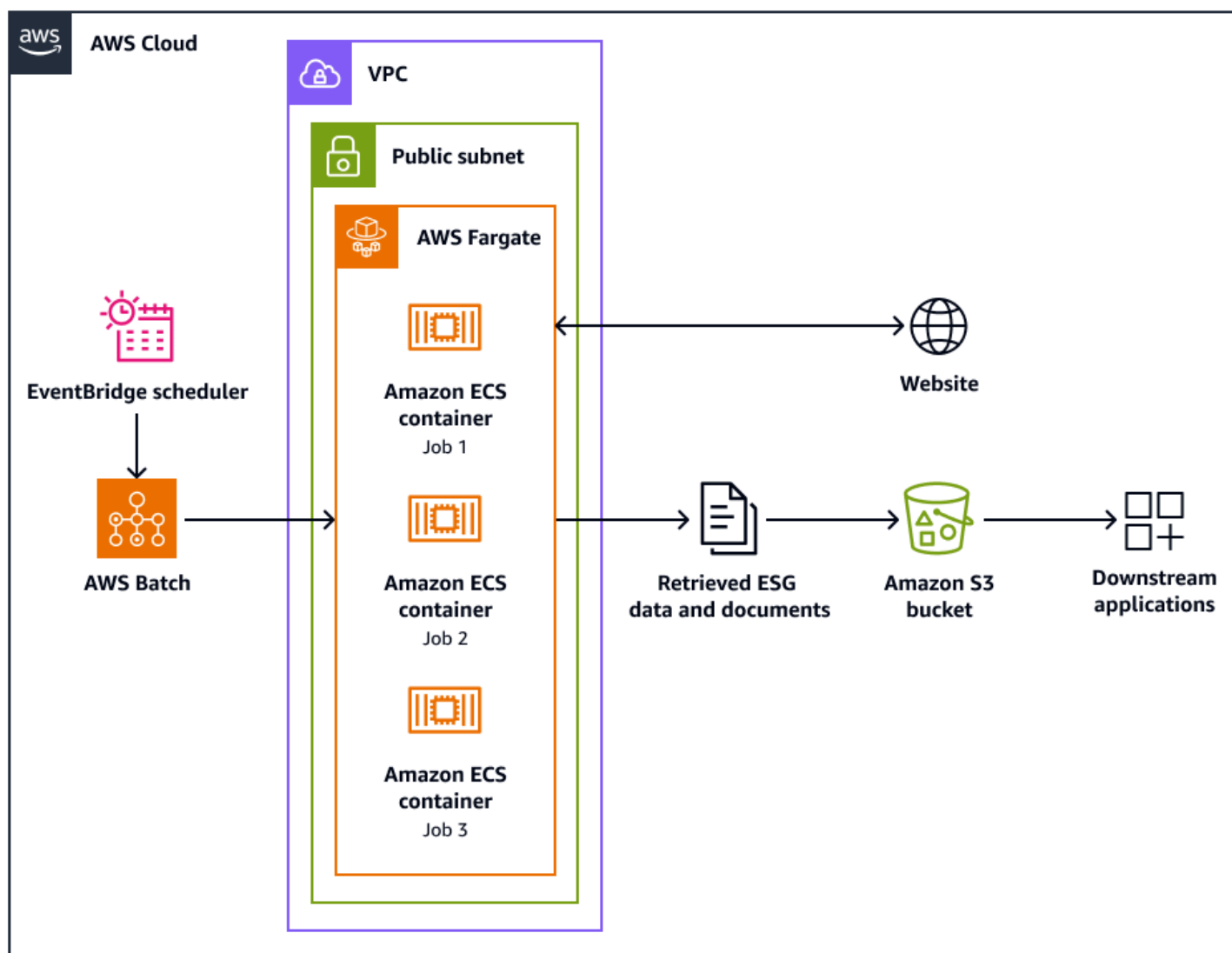
ターゲットを絞ったビジネス成果

企業が ESG データを使用する一般的な理由は次のとおりです。

- リスク管理 – ESG データは、環境、社会、ガバナンスの問題に関連する潜在的なリスクを特定して軽減するのに役立ちます。
- 投資家の誘致 — 多くの投資家は、投資を決定する際に ESG 要因を考慮しています。これらは、強力な ESG プラクティスを長期的な持続可能性と収益性の指標と見なしています。
- 評価管理 – 優れた ESG パフォーマンスは、顧客、従業員、一般の間で会社の評判を高めることができます。
- 規制コンプライアンス – ESG 関連の規制が増えるにつれて、企業がコンプライアンス要件を先取りするのに役立つ ESG プラクティスを採用します。
- イノベーションと効率性 – ESG 要因に焦点を当てることで、製品、サービス、運用におけるイノベーションを促進することができます。これにより、効率が向上し、コスト削減につながります。
- 競争上の利点 – 強力な ESG パフォーマンスは、企業を競合企業と差別化し、新しい市場機会を開くことができます。
- ステークホルダーのエンゲージメント – ESG プラクティスは、企業が従業員、顧客、地域コミュニティなど、さまざまなステークホルダーとより良く関わり、期待に応えるのに役立ちます。

でのスケーラブルなウェブクロールシステムのアーキテクチャ AWS

次のアーキテクチャ図は、ウェブサイトから環境、社会、ガバナンス (ESG) データを倫理的に抽出するように設計されたウェブクロールシステムを示しています。AWS インフラストラクチャ用に最適化された Python ベースのクローラを使用します。AWS Batch を使用して大規模なクロールジョブをオーケストレーションし、ストレージに Amazon Simple Storage Service (Amazon S3) を使用します。ダウンストリームアプリケーションは、Amazon S3 バケットからデータを取り込んで保存できます。



この図表は、次のワークフローを示しています:

1. Amazon EventBridge スケジューラは、スケジュールされた間隔でクローリングプロセスを開始します。
2. AWS Batch は、ウェブクロールジョブの実行を管理します。AWS Batch ジョブキューは、保留中のクローリングジョブを保持およびオーケストレーションします。
3. ウェブクロールジョブは、Amazon Elastic Container Service (Amazon ECS) コンテナで実行されます AWS Fargate。ジョブは、Virtual Private Cloud (VPC) のパブリックサブネットで行われます。
4. ウェブクロールはターゲットウェブサイトをクローリングし、PDF、CSV、またはその他のドキュメントファイルなどの ESG データおよびドキュメントを取得します。
5. ウェブクロールは、取得したデータと raw ファイルを Amazon S3 バケットに保存します。
6. 他のシステムまたはアプリケーションは、Amazon S3 バケットに保存されたデータとファイルを取り込み、処理します。

ウェブクロールの設計と運用

一部のウェブサイトは、デスクトップまたはモバイルデバイスで実行するように特別に設計されています。ウェブクロールは、デスクトップユーザーエージェントまたはモバイルユーザーエージェントの使用をサポートするように設計されています。これらのエージェントは、ターゲットウェブサイトへのリクエストを正常に行うのに役立ちます。

ウェブクロールが初期化されると、次のオペレーションが実行されます。

1. ウェブクロールは `setup()` メソッドを呼び出します。このメソッドは、`robots.txt` ファイルを取得して解析します。

Note

サイトマップを取得して解析するようにウェブクロールを設定することもできます。

2. ウェブクロールは `robots.txt` ファイルを処理します。`robots.txt` ファイルでクローリング遅延が指定されている場合、ウェブクロールはデスクトップユーザーエージェントのクローリング遅延を抽出します。`robots.txt` ファイルでクローリング遅延が指定されていない場合、ウェブクロールはランダムな遅延を使用します。
3. ウェブクロールは `crawl()` メソッドを呼び出し、クローリングプロセスを開始します。キューに URL がいない場合は、開始 URL を追加します。

Note

クローラは、最大ページ数に達するか、クローリングする URL が不足するまで続行されません。

4. クローラは URL をクローリングします。キュー内の各 URL について、クローラは URL がすでにクローリングされているかどうかを確認します。
5. URL がクローリングされていない場合、クローラは次のように `crawl_url()` メソッドを呼び出します。
 - a. クローラは `robots.txt` ファイルをチェックして、デスクトップユーザーエージェントを使用して URL をクローリングできるかどうかを判断します。
 - b. 許可されている場合、クローラはデスクトップユーザーエージェントを使用して URL のクローリングを試みます。
 - c. 許可されていない場合、またはデスクトップユーザーエージェントがクローリングに失敗した場合、クローラは `robots.txt` ファイルをチェックして、モバイルユーザーエージェントを使用して URL をクローリングできるかどうかを判断します。
 - d. 許可されている場合、クローラはモバイルユーザーエージェントを使用して URL のクローリングを試みます。
6. クローラは `attempt_crawl()` メソッドを呼び出し、コンテンツを取得して処理します。クローラは、適切なヘッダーを含む GET リクエストを URL に送信します。リクエストが失敗した場合、クローラは再試行ロジックを使用します。
7. ファイルが HTML 形式の場合、クローラは `extract_esg_data()` メソッドを呼び出します。[Beautiful Soup](#) を使用して HTML コンテンツを解析します。キーワードマッチングを使用して、環境、社会、ガバナンス (ESG) データを抽出します。

ファイルが PDF の場合、クローラは `save_pdf()` メソッドを呼び出します。クローラは PDF ファイルをダウンロードして Amazon S3 バケットに保存します。
8. クローラは `extract_news_links()` メソッドを呼び出します。これにより、ニュース記事、プレスリリース、ブログ投稿へのリンクが検索され、保存されます。
9. クローラは `extract_pdf_links()` メソッドを呼び出します。これにより、PDF ドキュメントへのリンクが識別され、保存されます。
10. クローラは `is_relevant_to_sustainable_finance()` メソッドを呼び出します。これにより、事前定義されたキーワードを使用して、ニュースまたは記事が持続可能な金融に関連しているかどうかを確認されます。

11. クローラが試行されるたびに、クローラは `delay()` メソッドを使用して遅延を実装します。robots.txt ファイルで遅延が指定されている場合、その値が使用されます。それ以外の場合は、1~3 秒のランダムな遅延を使用します。
12. クローラは、`save_esg_data()` メソッドを呼び出して ESG データを CSV ファイルに保存します。CSV ファイルは Amazon S3 バケットに保存されます。
13. クローラは `save_news_links()` メソッドを呼び出して、関連性情報を含む CSV ファイルにニュースリンクを保存します。CSV ファイルは Amazon S3 バケットに保存されます。
14. クローラは `save_pdf_links()` メソッドを呼び出して PDF リンクを CSV ファイルに保存します。CSV ファイルは Amazon S3 バケットに保存されます。

バッチ処理とデータ処理

クローリングプロセスは構造化された方法で整理および実行されます。AWS Batch は、ジョブを各会社のジョブを割り当てて、バッチで並行して実行します。各バッチは、データセットで特定した1つの会社のドメインとサブドメインに焦点を当てています。ただし、同じバッチ内のジョブは順番に実行されるため、リクエストが多すぎてウェブサイトに溢れません。これにより、アプリケーションはクローリングワークロードをより効率的に管理し、関連するすべてのデータを会社ごとに確実にキャプチャできます。

ウェブクローリングを会社固有のバッチに整理することで、収集されたデータをコンテナ化します。これにより、ある会社のデータが他の会社のデータと混在するのを防ぐことができます。

バッチ処理は、アプリケーションがウェブから効率的にデータを収集し、ターゲット企業とそれぞれのウェブドメインに基づいて情報の明確な構造と分離を維持するのに役立ちます。このアプローチは、収集されたデータの整合性と使いやすさを確保するのに役立ちます。これは、データが整理され、適切な会社とドメインに関連付けられているためです。

でのスケーラブルなウェブクロールシステムの構築

AWS

このセクションでは、[アーキテクチャ](#)セクションで説明されているウェブクローラーを構築する方法について説明します。これには、企業と関連するウェブプロパティの堅牢なデータセットを作成するための体系的なアプローチが含まれます。このデータセットは、クロールアクティビティの基盤として機能します。次に、このセクションでは、[倫理的ウェブクローラー](#)を構築する方法について説明しますPython。

トピック

- [データセットの準備](#)
- [ウェブクローラーの構築](#)
- [AWS インフラストラクチャの構築](#)

データセットの準備

まだ行っていない場合は、情報を収集するウェブサイトの詳細なデータセットを準備します。このデータセットには、ウェブサイト URL ドメイン名と関連するサブドメイン名を含める必要があります。このセクションでは、このデータセットを構築するためのstep-by-stepのプロセスについて説明します。

データセットを準備するには

1. 範囲を定義する – 重点を置く業界またはセクターを決定します。含める会社の数を決定します。また、従業員数、場所、収益など、これらの会社について収集する基準を定義します。
2. データソースの特定 – これらの会社に関する情報を収集するために使用できる情報源を特定します。例としては、ビジネスディレクトリ ([Crunchbase](#)、"[Forbes](#) など)、株式交換 (NYSE や NASDAQ など)、業界固有の関連付けや出版物、政府データベース (SEC ファイリングなど) などがあります。
3. テーブルの作成 – Microsoft Excel、Google Sheets、データベース管理システムなどの任意のツールで、各会社に関する基準を収集するためのテーブルを作成します。各条件の列を含めます。少なくとも、会社名、プライマリドメイン、サブドメイン、業界、サイズ、場所の列を含めます。
4. 初期会社情報を収集する – 各会社に関する次の情報を収集し、作成したテーブルに入力します。

- 会社名
 - 業界またはセクター
 - 会社規模 (従業員数)
 - Revenue
 - 本社の場所
5. ドメイン情報を収集する – 会社ごとに、などのメインウェブサイト URL からプライマリドメイン名を抽出します `example.com`。WHOIS ドメインルックアップツールを使用してドメイン情報を確認できます。
 6. サブドメイン情報を収集する – 会社ごとに、などの登録されたサブドメインを調べます `blog.example.com`。 [Sublist3r](#)、 [OWASP Amass](#)、 [Subfinder](#) などのサブドメイン列挙ツールを使用できます。Google のドッキング (を検索 `site:example.com`) を実行するか、 `dig` コマンドまたは DNS ルックアップツールを使用して DNS レコードを確認するか、SSL 証明書または TLS 証明書を分析できます。
 7. データを検証してクリーンアップする – 収集したデータを確認、検証、標準化します。たとえば、重複するエントリを削除し、ドメインとサブドメインから不要な URL 情報を削除して、すべてのドメインとサブドメインがアクティブであることを確認します。
 8. (オプション) サブドメインを分類する – サブドメインをタイプに分類できます。以下は、発生する可能性のあるカテゴリの例です。
 - などのブログ `blog.example.com`
 - や などのサポート `support.example.com` または ヘルプ `help.example.com`
 - `shop.example.com` や などの E コマース `store.example.com`
 - `dev.example.com` や などのデベロッパーリソース `api.example.com`
 - や などのリージョン `us.example.com` または ロケーション `uk.example.com`
 9. (オプション) 関連するメタデータを追加する – 関連するメタデータをデータセットに記録できます。たとえば、最終更新日、情報源、サブドメインの精度の信頼スコアを追加できます。
 10. バージョン管理の実装 – Git などのバージョン管理システムを使用して、テーブルへの経時的な変更を追跡します。データセットを定期的にバックアップします。
 11. テーブルの維持 – テーブルを更新するための四半期ごとのスケジュールを設定します。新しい会社を追加したり、不要になった会社を削除したりするプロセスを標準化して実装します。可能であれば、サブドメインの検出を自動化します。

ウェブクローラーの構築

[アーキテクチャ](#) 「」セクションで説明されているように、アプリケーションは会社ごとに 1 つずつバッチで実行されます。

トピック

- [robots.txt ファイルのキャプチャと処理](#)
- [サイトマップのキャプチャと処理](#)
- [クローラの設計](#)

robots.txt ファイルのキャプチャと処理

データセットを準備したら、ドメインに [robots.txt](#) ファイルがあるかどうかを確認する必要があります。ウェブクローラーやその他のボットの場合、robots.txt ファイルはウェブサイトのどのセクションにアクセスできるかを示します。このファイルの指示に従うことは、ウェブサイトを倫理的にクローリングするための重要なベストプラクティスです。詳細については、このガイドの [「倫理的ウェブクローラーのベストプラクティス」](#) を参照してください。

robots.txt ファイルをキャプチャして処理するには

1. まだインストールしていない場合は、ターミナルで次のコマンドを実行して requests ライブラリをインストールします。

```
pip install requests
```

2. 次のスクリプトを実行します。このスクリプトは以下の処理を実行します:

- ドメインを入力として受け取る check_robots_txt 関数を定義します。
- robots.txt ファイルの完全な URL を作成します。
- robots.txt ファイルの URL に GET リクエストを送信します。
- リクエストが成功した場合 (ステータスコード 200)、robots.txt ファイルがあります。
- リクエストが失敗するか、別のステータスコードを返す場合、robots.txt ファイルが存在しないか、アクセスできません。

```
import requests
from urllib.parse import urljoin
```

```
def check_robots_txt(domain):
    # Ensure the domain starts with a protocol
    if not domain.startswith(('http://', 'https://')):
        domain = 'https://' + domain
    # Construct the full URL for robots.txt
    robots_url = urljoin(domain, '/robots.txt')
    try:
        # Send a GET request to the robots.txt URL
        response = requests.get(robots_url, timeout=5)
        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            print(f"robots.txt found at {robots_url}")
            return True
        else:
            print(f"No robots.txt found at {robots_url} (Status code:
{response.status_code})")
            return False
    except requests.RequestException as e:
        print(f"Error checking {robots_url}: {e}")
        return False
```

Note

このスクリプトは、ネットワークエラーやその他の問題の例外を処理します。

3. robots.txt ファイルが存在する場合は、次のスクリプトを使用してダウンロードします。

```
import requests

def download(self, url):
    response = requests.get(url, headers=self.headers, timeout=5)
    response.raise_for_status() # Raise an exception for non-2xx responses
    return response.text

def download_robots_txt(self):
    # Append '/robots.txt' to the URL to get the robots.txt file's URL
    robots_url = self.url.rstrip('/') + '/robots.txt'
    try:
        response = download(robots_url)
        return response
    except requests.exceptions.RequestException as e:
```

```
print(f"Error downloading robots.txt: {e}, \nGenerating sitemap using combinations...")
return e
```

Note

これらのスクリプトは、ユースケースに応じてカスタマイズまたは変更できます。これらのスクリプトを組み合わせることもできます。

サイトマップのキャプチャと処理

次に、[サイトマップ](#)を処理する必要があります。サイトマップを使用して、重要なページのクロールに集中できます。これにより、クローリングの効率が向上します。詳細については、このガイドの「[倫理的ウェブクロウラーのベストプラクティス](#)」を参照してください。

サイトマップをキャプチャして処理するには

- 次のスクリプトを実行します。このスクリプトは、以下の`check_and_download_sitemap`関数を定義します。
 - ベース URL、robots.txt からのオプションのサイトマップ URL、およびユーザーエージェント文字列を受け入れます。
 - robots.txt (提供されている場合) からのサイトマップを含む、複数の潜在的なサイトマップの場所をチェックします。
 - 各場所からサイトマップをダウンロードしようとします。
 - ダウンロードしたコンテンツが XML 形式であることを確認します。
 - `parse_sitemap` 関数を呼び出して URLs。この関数:
 - サイトマップの XML コンテンツを解析します。
 - 通常のサイトマップとサイトマップインデックスファイルの両方を処理します。
 - サイトマップインデックスが発生した場合、サブサイトマップを再帰的に取得します。

```
import requests
from urllib.parse import urljoin
import xml.etree.ElementTree as ET
```

```
def check_and_download_sitemap(base_url, robots_sitemap_url=None,
    user_agent='SitemapBot/1.0'):
    headers = {'User-Agent': user_agent}
    sitemap_locations = [robots_sitemap_url, urljoin(base_url, '/sitemap.xml'),
        urljoin(base_url, '/sitemap_index.xml'),
        urljoin(base_url, '/sitemap/'), urljoin(base_url, '/sitemap/sitemap.xml')]

    for sitemap_url in sitemap_locations:
        if not sitemap_url:
            continue
        print(f"Checking for sitemap at: {sitemap_url}")
        try:
            response = requests.get(sitemap_url, headers=headers, timeout=10)
            if response.status_code == 200:
                content_type = response.headers.get('Content-Type', '')
                if 'xml' in content_type:
                    print(f"Successfully downloaded sitemap from {sitemap_url}")
                    return parse_sitemap(response.text)
                else:
                    print(f"Found content at {sitemap_url}, but it's not XML.
Content-Type: {content_type}")
            except requests.RequestException as e:
                print(f"Error downloading sitemap from {sitemap_url}: {e}")
        print("No sitemap found.")
    return []

def parse_sitemap(sitemap_content):
    urls = []
    try:
        root = ET.fromstring(sitemap_content)
        # Handle both sitemap and sitemapindex
        for loc in root.findall('.://{http://www.sitemaps.org/schemas/
sitemap/0.9}loc'):
            urls.append(loc.text)

        # If it's a sitemap index, recursively fetch each sitemap
        if root.tag.endswith('sitemapindex'):
            all_urls = []
            for url in urls:
                print(f"Fetching sub-sitemap: {url}")
                sub_sitemap_urls = check_and_download_sitemap(url)
                all_urls.extend(sub_sitemap_urls)
            return all_urls
    except ET.ParseError as e:
```

```
        print(f"Error parsing sitemap XML: {e}")
    return urls

if __name__ == "__main__":
    base_url = input("Enter the base URL of the website: ")
    robots_sitemap_url = input("Enter the sitemap URL from robots.txt (or press
Enter if none): ").strip() or None
    urls = check_and_download_sitemap(base_url, robots_sitemap_url)
    print(f"Found {len(urls)} URLs in sitemap:")
    for url in urls[:5]: # Print first 5 URLs as an example
        print(url)
    if len(urls) > 5:
        print("...")
```

クローラの設計

次に、ウェブクローラーを設計します。クローラは、このガイド[倫理的ウェブクローラーのベストプラクティス](#)で説明されているベストプラクティスに従うように設計されています。この EthicalCrawler クラスは、倫理的クローラのいくつかの主要な原則を示しています。

- robots.txt ファイルの取得と解析 — クローラはターゲットウェブサイトの robots.txt ファイルを取得します。
- クローリング許可の尊重 – URL をクローリングする前に、クローラは robots.txt ファイルのルールでその URL のクローリングが許可されているかどうかを確認します。URL が許可されていない場合、クローラはその URL をスキップして次の URL に移動します。
- クロール遅延の尊重 – クローラは robots.txt ファイル内のクロール遅延ディレクティブをチェックします。指定した場合、クローラはリクエスト間でこの遅延を使用します。それ以外の場合は、デフォルトの遅延を使用します。
- ユーザーエージェント識別 – クローラはカスタムユーザーエージェント文字列を使用して、ウェブサイトに対して自身を識別します。必要に応じて、ウェブサイトの所有者は、クローラを制限または許可する特定のルールを設定できます。
- エラー処理と正常な低下 — robots.txt ファイルをフェッチまたは解析できない場合、クローラは保守的なデフォルトルールに進みます。ネットワークエラーと 200 以外の HTTP レスポンスを処理します。
- 制限付きクローリング – サーバーの負担を避けるために、クローリングできるページ数に制限があります。

次のスクリプトは、ウェブクローラーの仕組みを説明する擬似コードです。

```
import requests
from urllib.parse import urljoin, urlparse
import time

class EthicalCrawler:
    def __init__(self, start_url, user_agent='EthicalBot/1.0'):
        self.start_url = start_url
        self.user_agent = user_agent
        self.domain = urlparse(start_url).netloc
        self.robots_parser = None
        self.crawl_delay = 1 # Default delay in seconds

    def can_fetch(self, url):
        if self.robots_parser:
            return self.robots_parser.allowed(url, self.user_agent)
        return True # If no robots.txt, assume allowed but crawl conservatively

    def get_crawl_delay(self):
        if self.robots_parser:
            delay = self.robots_parser.agent(self.user_agent).delay
            if delay is not None:
                self.crawl_delay = delay
        print(f"Using crawl delay of {self.crawl_delay} seconds")

    def crawl(self, max_pages=10):
        self.get_crawl_delay()
        pages_crawled = 0
        urls_to_crawl = [self.start_url]
        while urls_to_crawl and pages_crawled < max_pages:
            url = urls_to_crawl.pop(0)
            if not self.can_fetch(url):
                print(f"robots.txt disallows crawling: {url}")
                continue
            try:
                response = requests.get(url, headers={'User-Agent': self.user_agent})
                if response.status_code == 200:
                    print(f"Successfully crawled: {url}")
                    # Here you would typically parse the content, extract links, etc.
                    # For this example, we'll just increment the counter
                    pages_crawled += 1
            else:
                print(f"Failed to crawl {url}: HTTP {response.status_code}")
```

```
except Exception as e:
    print(f"Error crawling {url}: {e}")

# Respect the crawl delay
time.sleep(self.crawl_delay)

print(f"Crawling complete. Crawled {pages_crawled} pages.")
```

ESG データを収集する高度な倫理的ウェブクローラーを構築するには

1. このシステムで使用される高度な倫理的ウェブクローラーの次のコードサンプルをコピーします。

```
import requests
from urllib.parse import urljoin, urlparse
import time
from collections import deque
import random
from bs4 import BeautifulSoup
import re
import csv
import os

class EnhancedESGCrawler:
    def __init__(self, start_url):
        self.start_url = start_url
        self.domain = urlparse(start_url).netloc
        self.desktop_user_agent = 'ESGEthicalBot/1.0'
        self.mobile_user_agent = 'Mozilla/5.0 (iPhone; CPU iPhone OS 14_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/15E148 Safari/604.1'
        self.robots_parser = None
        self.crawl_delay = None
        self.urls_to_crawl = deque()
        self.crawled_urls = set()
        self.max_retries = 2
        self.session = requests.Session()
        self.esg_data = []
        self.news_links = []
        self.pdf_links = []

    def setup(self):
```

```
self.fetch_robots_txt() # Provided in Previous Snippet
self.fetch_sitemap() # Provided in Previous Snippet

def can_fetch(self, url, user_agent):
    if self.robots_parser:
        return self.robots_parser.allowed(url, user_agent)
    return True

def delay(self):
    if self.crawl_delay is not None:
        time.sleep(self.crawl_delay)
    else:
        time.sleep(random.uniform(1, 3))

def get_headers(self, user_agent):
    return {'User-Agent': user_agent,
            'Accept': 'text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8',
            'Accept-Language': 'en-US,en;q=0.5', 'Accept-Encoding': 'gzip,
deflate, br', 'DNT': '1',
            'Connection': 'keep-alive', 'Upgrade-Insecure-Requests': '1'}

def extract_esg_data(self, url, html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    esg_data = {
        'url': url,
        'environmental': self.extract_environmental_data(soup),
        'social': self.extract_social_data(soup),
        'governance': self.extract_governance_data(soup)
    }
    self.esg_data.append(esg_data)
    # Extract news links and PDFs
    self.extract_news_links(soup, url)
    self.extract_pdf_links(soup, url)

def extract_environmental_data(self, soup):
    keywords = ['carbon footprint', 'emissions', 'renewable energy', 'waste
management', 'climate change']
    return self.extract_keyword_data(soup, keywords)

def extract_social_data(self, soup):
    keywords = ['diversity', 'inclusion', 'human rights', 'labor practices',
'community engagement']
    return self.extract_keyword_data(soup, keywords)
```

```
def extract_governance_data(self, soup):
    keywords = ['board structure', 'executive compensation', 'shareholder
rights', 'ethics', 'transparency']
    return self.extract_keyword_data(soup, keywords)

def extract_keyword_data(self, soup, keywords):
    text = soup.get_text().lower()
    return {keyword: len(re.findall(r'\b' + re.escape(keyword) + r'\b', text))
for keyword in keywords}

def extract_news_links(self, soup, base_url):
    news_keywords = ['news', 'press release', 'article', 'blog',
'sustainability']
    for a in soup.find_all('a', href=True):
        if any(keyword in a.text.lower() for keyword in news_keywords):
            full_url = urljoin(base_url, a['href'])
            if full_url not in self.news_links:
                self.news_links.append({'url': full_url, 'text':
a.text.strip()})

def extract_pdf_links(self, soup, base_url):
    for a in soup.find_all('a', href=True):
        if a['href'].lower().endswith('.pdf'):
            full_url = urljoin(base_url, a['href'])
            if full_url not in self.pdf_links:
                self.pdf_links.append({'url': full_url, 'text':
a.text.strip()})

def is_relevant_to_sustainable_finance(self, text):
    keywords = ['sustainable finance', 'esg', 'green bond', 'social impact',
'environmental impact',
                'climate risk', 'sustainability report', 'corporate
responsibility']
    return any(keyword in text.lower() for keyword in keywords)

def attempt_crawl(self, url, user_agent):
    for _ in range(self.max_retries):
        try:
            response = self.session.get(url,
headers=self.get_headers(user_agent), timeout=10)
            if response.status_code == 200:
                print(f"Successfully crawled: {url}")
```

```
        if response.headers.get('Content-Type', '').startswith('text/html'):
            self.extract_esg_data(url, response.text)
        elif response.headers.get('Content-Type', '').startswith('application/pdf'):
            self.save_pdf(url, response.content)
            return True
        else:
            print(f"Failed to crawl {url}: HTTP {response.status_code}")
    except requests.RequestException as e:
        print(f"Error crawling {url} with {user_agent}: {e}")

    self.delay()
    return False

def crawl_url(self, url):
    if not self.can_fetch(url, self.desktop_user_agent):
        print(f"Robots.txt disallows desktop user agent: {url}")
    if self.can_fetch(url, self.mobile_user_agent):
        print(f"Attempting with mobile user agent: {url}")
        return self.attempt_crawl(url, self.mobile_user_agent)
    else:
        print(f"Robots.txt disallows both user agents: {url}")
        return False

    return self.attempt_crawl(url, self.desktop_user_agent)

def crawl(self, max_pages=100):
    self.setup()

    if not self.urls_to_crawl:
        self.urls_to_crawl.append(self.start_url)

    pages_crawled = 0
    while self.urls_to_crawl and pages_crawled < max_pages:
        url = self.urls_to_crawl.popleft()
        if url not in self.crawled_urls:
            if self.crawl_url(url):
                pages_crawled += 1
                self.crawled_urls.add(url)
            self.delay()

    print(f"Crawling complete. Successfully crawled {pages_crawled} pages.")
    self.save_esg_data()
```

```
self.save_news_links()
self.save_pdf_links()

def save_esg_data(self):
    with open('esg_data.csv', 'w', newline='', encoding='utf-8') as file:
        writer = csv.DictWriter(file, fieldnames=['url', 'environmental',
'social', 'governance'])
        writer.writeheader()
        for data in self.esg_data:
            writer.writerow({
                'url': data['url'],
                'environmental': ', '.join([f"{k}: {v}" for k, v in
data['environmental'].items()]),
                'social': ', '.join([f"{k}: {v}" for k, v in
data['social'].items()]),
                'governance': ', '.join([f"{k}: {v}" for k, v in
data['governance'].items()])
            })
        print("ESG data saved to esg_data.csv")

def save_news_links(self):
    with open('news_links.csv', 'w', newline='', encoding='utf-8') as file:
        writer = csv.DictWriter(file, fieldnames=['url', 'text', 'relevant'])
        writer.writeheader()
        for news in self.news_links:
            writer.writerow({
                'url': news['url'],
                'text': news['text'],
                'relevant':
self.is_relevant_to_sustainable_finance(news['text'])
            })
        print("News links saved to news_links.csv")

def save_pdf_links(self):
    # Code for saving PDF in S3 or filesystem

def save_pdf(self, url, content):
    # Code for saving PDF in S3 or filesystem

# Example usage
if __name__ == "__main__":
    start_url = input("Enter the starting URL to crawl for ESG data and news: ")
    crawler = EnhancedESGCrawler(start_url)
```

```
crawler.crawl(max_pages=50)
```

2. ユーザーエージェント、URLs の空のコレクション、データストレージリストなど、さまざまな属性を設定します。
3. `is_relevant_to_sustainable_finance()` メソッドのキーワードと関連性基準を、特定のニーズに合わせて調整します。
4. `robots.txt` ファイルがウェブサイトのクローリングを許可し、`robots.txt` ファイルで指定されたクローリング遅延とユーザーエージェントを使用していることを確認します。
5. 組織が必要に応じて、提供されたウェブクローラースクリプトに次のカスタマイズを行うことを検討してください。
 - より効率的な URL 検出のために `fetch_sitemap()` メソッドを実装します。
 - 本番稼働用のエラーログ記録と処理を強化します。
 - より高度なコンテンツ関連性分析を実装します。
 - 深さと幅のコントロールを追加して、クローリング範囲を制限します。

AWS インフラストラクチャの構築

ウェブクローリングインフラストラクチャの構築 AWS のサービス には使用できるものは多数あります。このガイドの [アーキテクチャ](#) セクションには、提案されたソリューションが 1 つ含まれています。ウェブクローラーのサポートインフラストラクチャを構築 AWS のサービス するために、以下を使用することを検討することをお勧めします。

- Amazon Virtual Private Cloud (Amazon VPC) を使用して [VPC とサブネット](#) を作成します。
- [Amazon EventBridge スケジューラ](#) を使用してクローリングプロセスを開始します。
- AWS Batch [ジョブ](#) と [ジョブキュー](#) を使用して [ウェブクローラージョブ](#) を管理します。
- 次のいずれかのソリューションを使用して、ウェブクローラージョブを実行します。
 - の Amazon Elastic Container Service (Amazon ECS) コンテナ [AWS Fargate](#)
 - Amazon Elastic Compute Cloud (Amazon EC2) [インスタンス](#)

Note

アプリケーションで中断を処理できる場合は、スポット [フリート](#) 経由で Amazon EC2 [スポットインスタンス](#) を使用することを検討してください。スポットインスタンスのフリートは、コンピューティングコストを大幅に削減するのに役立ちます。

-
- AWS Lambda [関数](#)
 - 取得したデータと raw ファイルを Amazon Simple Storage Service (Amazon S3) [バケット](#) に保存します。

倫理的ウェブクローラーのベストプラクティス

このセクションでは、環境、社会、ガバナンス (ESG) データを収集するウェブクローリングアプリケーションを構築するためのベストプラクティスと重要な倫理的考慮事項について説明します。これらのベストプラクティスに従うことで、プロジェクトと組織を保護し、より責任ある持続可能なウェブエコシステムに貢献できます。このアプローチは、貴重なデータにアクセスし、すべての利害関係者を尊重する方法で研究、ビジネス、イノベーションに使用するのに役立ちます。

Robots.txt コンプライアンス

robots.txt ファイルは、ウェブサイトで使用され、ウェブサイトのどの部分にアクセスまたはクローリングすべきかについて、ウェブクローラーやボットと通信します。ウェブクローラーは、ウェブサイトでの robots.txt ファイルを検出すると、指示を解析し、それに応じてクローリング動作を調整します。これにより、クローラーがウェブサイト所有者の指示に違反しないようにし、ウェブサイトとクローラーの間の協力関係を維持します。したがって、robots.txt ファイルは、アクセスコントロール、機密コンテンツの保護、ロード管理、および法的コンプライアンスに役立ちます。

ACM のベストプラクティスに従うことをおすすめします。

- robots.txt ファイルのルールを常に確認し、尊重してください。
- URL をクローリングする前に、デスクトップユーザーエージェントとモバイルユーザーエージェントの両方のルールを確認してください。
- ウェブサイトでモバイルユーザーエージェントのみが許可されている場合は、リクエストにモバイルユーザーエージェントヘッダーなどの別のユーザーエージェントヘッダーを使用します。

robots.txt ファイルがないからといって、ウェブサイトをクローリングできない、またはクローリングすべきではないとは限りません。クローリングは、ウェブサイトのリソースと所有者の暗黙的な権利を尊重して、常に責任を持って行う必要があります。robots.txt が存在しない場合に推奨されるベストプラクティスは次のとおりです。

- クローリングは許可されていると仮定しますが、注意して続行してください。
- 礼儀正しいクローリングプラクティスを実装します。
- 大規模なクローリングを実行する予定がある場合は、ウェブサイト所有者に連絡して許可を求めることを検討してください。

クローリングレートの制限

サーバーが過負荷にならないように、適切なクローリングレートを使用します。robots.txt ファイルで指定されているか、ランダムな遅延を使用して、リクエスト間の遅延を実装します。小規模または中規模のウェブサイトでは、10～15 秒ごとに 1 つのリクエストが適切である場合があります。大規模なウェブサイトや明示的なクローリング許可を持つウェブサイトでは、1 秒あたり 1～2 リクエストが適切である場合があります。

ユーザーエージェントの透明性

user-agent ヘッダーでクローラを特定します。この HTTP ヘッダー情報は、コンテンツをリクエストしているデバイスを識別することを目的としています。通常、ボットという単語はエージェントの名前に含まれます。クローラやその他のボットは、ヘッダーの重要なフィールドを使用して連絡先情報を含めることがあります。

効率的なクローリング

ウェブサイト所有者によって開発されたサイトマップを使用して、重要なページに焦点を当てます。

アダプティブアプローチ

デスクトップバージョンが失敗した場合にモバイルユーザーエージェントに切り替えるようにクローラをプログラムします。これにより、クローラへのアクセスが可能になり、ウェブサイトのサーバーへの負担を軽減できます。

エラー処理

クローラがさまざまな HTTP ステータスコードを適切に処理していることを確認します。たとえば、429 ステータスコード（「リクエストが多すぎます」）が発生した場合、クローラは一時停止する必要があります。クローラが 403 ステータスコード（「禁止」）を継続的に受信する場合は、クローリングの停止を検討してください。

バッチでのクローリング

次のことを行うことをお勧めします。

- すべての URLs 一度にクローリングする代わりに、タスクを小さなバッチに分割します。これにより、負荷を分散し、タイムアウトやリソース制約などの問題が発生するリスクを軽減できます。
- クローリングタスク全体が長時間実行されることが予想される場合は、より小さく管理しやすい複数のタスクに分割することを検討してください。これにより、プロセスのスケーラビリティと耐障害性が向上します。
- クローリングする URLs の数が比較的少ない場合は、などのサーバーレスソリューションの使用を検討してください AWS Lambda。Lambda 関数は、リソース管理を自動的にスケーリングして処理するため、存続期間の短いイベント駆動型タスクに適しています。

セキュリティ

ウェブクローリングコンピューティングタスクでは、アウトバウンドトラフィックのみを許可するように環境を設定することをお勧めします。これにより、攻撃対象領域を最小限に抑え、不正なインバウンドアクセスのリスクを減らすことで、セキュリティを強化できます。アウトバウンド接続のみを許可すると、クローリングプロセスがターゲットウェブサイトと通信して必要なデータを取得でき、システムが侵害される可能性のあるインバウンドトラフィックが制限されます。

その他の考慮事項

以下の追加の考慮事項とベストプラクティスを確認してください。

- ウェブサイトの利用規約またはプライバシーポリシーのクローリングガイドラインを確認してください。
- HTML で、クローリングディレクティブを提供する可能性のある meta タグを探します。
- データ収集と使用に関する管轄区域の法的制限に注意してください。
- ウェブサイト所有者から要求された場合は、クローリングを停止する準備をしてください。

よくある質問

robots.txt ファイルが使用できない場合はどうなりますか？

robots.txt ファイルがないからといって、ウェブサイトをクローリングできない、またはクローリングすべきではないとは限りません。クローリングは、ウェブサイトのリソースとウェブサイト所有者の暗黙的な権利を尊重し、常に責任を持って行う必要があります。

sitemaps.xml ファイルが使用できない場合はどうなりますか？

要件に応じて、次のいずれかを実行できます。

- HTML サイトマップの検索 – ウェブサイトの重要なページを一覧表示する HTML サイトマップ ページを探します。これらはフッターにリンクされることがよくあります。
- ホームページからのクローリング – ホームページからのクローリングを開始し、内部リンクに従って他のページを検出します。
- URL パターンの分析 – ウェブサイトの URL 構造を分析してパターンを特定し、潜在的な URLs。
- robots.txt ファイルを確認する – robots.txt ファイルで、許可されていないページやディレクトリがないかどうかを確認します。これらは、サイト構造に関する手がかりを提供することができます。
- API エンドポイントの確認 – 一部のウェブサイトでは、コンテンツと構造情報の取得に使用できる API エンドポイントを提供しています。
- 検索エンジンの結果を確認する – 検索エンジンを使用して、`site:` などの[検索演算子](#)を使用してウェブサイトのインデックス付きページを検索します `site:example.com`。
- バックリンクを分析する – ウェブサイトへのバックリンクを分析して、他のサイトがリンクしている重要なページを見つけます。
- ウェブアーカイブの確認 – [Wayback Machine](#) などのインターネットアーカイブで、サイトマップや異なる構造を持つ可能性のある古いバージョンのサイトを確認します。
- コンテンツ管理システム (CMS) パターンを探す – CMS を特定できる場合は、そのシステムに関連付けられた一般的な URL パターンを使用します。
- JavaScript レンダリングの確認 – サイトが JavaScript に大きく依存している場合は、クローラーが JavaScript をレンダリングして動的にロードされたコンテンツを検出できることを確認してください。一部のウェブサイトでは、JavaScript レンダリングが有効になった後に `sitemap.xml` ファイルがロードされます。

Amazon EC2 または Amazon ECS の代わりにサーバーレスソリューションを使用できますか？

はい。ウェブクローリングの [AWS Lambda](#) 関数は、特に小規模またはよりモジュール式のクローリングタスクの場合、実行可能なオプションです。ただし、大規模で長時間実行されるクローリングオペレーションでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスまたは Amazon Elastic Container Service (Amazon ECS) を使用する従来のアプローチの方が適している場合があります。ウェブクローリングのニーズに適したコンピューティングサービスを選択するときは、特定の要件とトレードオフを慎重に評価することが重要です。

クローラが 403 ステータスコードを取得するのはなぜですか？

HTTP 403 は、リクエストされたリソースへのアクセスが禁止される HTTP ステータスコードです。リクエストが正しい場合、サーバーはリクエストを理解し、それを達成しません。403 ステータスコードを防ぐには、次の操作を行います。

- クロールレートを制限します。
- sitemap または robots.txt ファイルで、クローラが URL にアクセスできるかどうかを確認します。
- デスクトップユーザーエージェントではなくモバイルユーザーエージェントで試してください。

上記のいずれも機能しない場合は、ウェブサイト所有者の決定を尊重し、ページをクローリングしないでください。

次のステップとリソース

raw 環境、社会、ガバナンス (ESG) データを収集したら、以下を実行してデータから意味のある情報を抽出できます。

1. データのクリーンアップ – データには、ESG 要因や財務データとは無関係な大量の無関係な情報が含まれている場合があります。この無関係なデータを削除し、必要な分析を実行するために必要な情報のみを保持することが重要です。[yfinance](#) などのツールを使用して、データをクリーンアップできます。
2. データを抽出して変換する – raw データから関連する特徴量や変数を抽出し、分析に適した形式に変換します。読みやすくわかりやすいように、データを表形式に変換できます。などのライブラリを使用して [pandas](#) データを絞り込むことができます。特徴量エンジニアリング、データ正規化、派生メトリクスを使用してデータを変換することもできます。
3. 分析の実行 – さまざまな分析タスクを実行できます。これには、記述統計の生成、データの視覚化の作成、企業の ESG パフォーマンスに関するインサイトを取得するための探索的データ分析の実施などが含まれます。
4. 機械学習を適用する – クリーンアップおよび変換されたデータを使用して、機械学習モデルをトレーニングできます。これらのモデルは、現在財務上の持続可能性を実践している企業を特定し、将来の持続可能性パフォーマンスを予測するのに役立ちます。

ウェブクローラーとこのデータ評価プロセスを使用することで、評価している企業の持続可能性のプラクティスと財務パフォーマンスを効果的に理解できます。この情報を使用して、投資決定の通知、進捗状況の追跡、持続可能なビジネスプラクティスのサポートを行うことができます。

リソース

- [ウェブクローラーとは](#) (ウェブサイト) Cloudflare
- [ESG 投資ガイド](#) (Investopedia ウェブサイト)

ツール

- [Beautiful Soup](#) (Beautiful Soup ドキュメント)
- [表形式のリレーショナルデータの処理](#) (pandas ウェブサイト)

ドキュメント履歴

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

変更	説明	日付
初版発行	—	2025 年 1 月 6 日

AWS 規範ガイドの用語集

以下は、AWS 規範ガイドが提供する戦略、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。

アベイラビリティゾーン (AZ)

他のアベイラビリティゾーンの障害から 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 するための、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一つ) 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なら、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。

ブレイクグラスアクセス

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 ガイドの「[ブレイ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。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で利用できるサービス](#)」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 (ポストン: 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利用できる、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

laC

「[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。

I

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/サブスクリプション](#) 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 内のすべてのイベント AWS CloudTrail をログに記録すること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ン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

読み取り専用で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)

サービスが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。

レスポンスコントロール

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「[レスポンスコントロール](#)」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](#)、[レスポンス](#)、[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

のエンドポイントの 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が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と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「[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。