CI/CD リトマステスト: パイプラインは完全に CI/CD ですか？

AWS 規範ガイダンス

AWS 規範ガイダンス: CI/CD リトマステスト: パイプラインは完全に CI/CD ですか？

Table of Contents

CI/CD リトムテスト: パイプラインは完全に CI/CD ですか？

Steven Guggenheimer and Ananya Koduri、Amazon Web Services (AWS)

2023 年 8 月 (ドキュメント履歴)

パイプラインは自動化されていますか？ これは簡単な質問ですが、多くの組織は単純に答えにアプローチしすぎます。答えは、はいまたはいいえよりもはるかに複雑です。

テクノロジーのイノベーションは絶えず起こり、組織が遅れないようにするのは難しい場合があります。この新しいモノはフェードなのか、それとも次の大きなモノなのか。現在のプラクティスを見直したり、待つべきか。多くの場合、何かが実際に次の大きなものであることが明確になるまでに、キャッチアップをプレイしていることがわかります。継続的インテグレーションと継続的デリバリー (CI/CD) は維持するためにここにありますが、必ずしもそうではありませんでした。多くの人が確信するのに長い時間がかかり、一部の人はまだより確信する必要があります。

CI/CD は、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、本番稼働の各段階を自動化するプロセスであり、一般的にパイプラインと呼ばれます。現在、CI/CD オートメーションのコスト削減とスピードにより、ほとんどの組織がその価値を確信しています。しかし、この新しいアプローチへの移行は簡単な作業ではありません。スタッフに適切なトレーニングがあることを確認し、一部のリソースをアップグレードしてから、テスト、テスト、テストする必要があります。やるべきことがたくさんあります。ほとんどの場合、組織が適応できるように、これらの変更を徐々に加えたいと考えています。

このドキュメントの目的は、完全な CI/CD プロセスを持つことの意味を定義することです。独自のプロセスを評価するツールを提供し、まだ存在しないプロセスの進路を提示します。このパスフォワードが夜間変換になることはほとんどありません。これらのプロセスは複雑で、現在の従業員のスキルセットや現在のインフラストラクチャ需要など、多くの要因に依存します。優先順位を付け、小規模で段階的な変更を行うことをお勧めします。

目的

このガイドのレコメンデーションを実装すると、次のような利点があります。

- 効率 – 完全な CI/CD デプロイプロセスにより、デバッグ、手動プロセスの実行、保守に費やされた複雑さ、ワークロード、無数の時間を削減できます。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。TechAhead ブログ記事によると、CI/CD プロセスの実装により、推定 20% の時間、労力、リソースを節約できます。

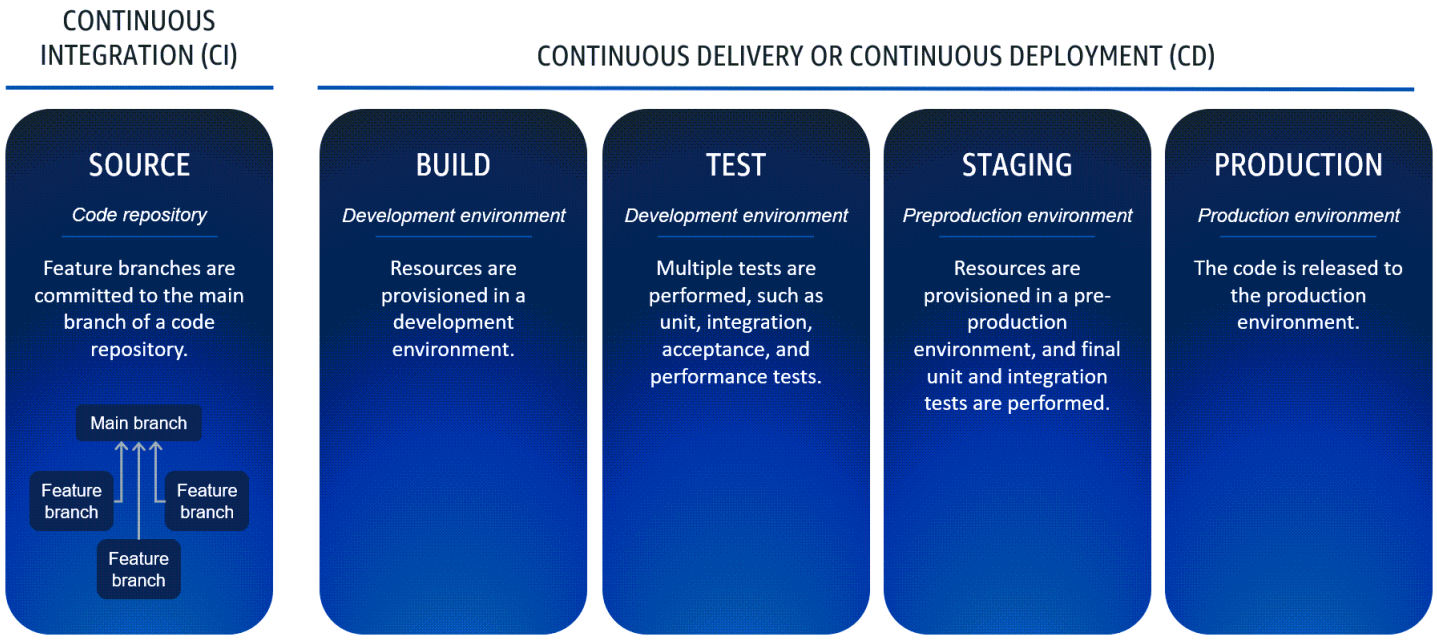
- コスト削減 — Forbes Insight レポートによると、「4 人のエグゼクティブのうち 3 人は、継続的なメンテナンスと管理に費やされた時間、資金、リソースが、新しいプロジェクト開発や新しいイニシアチブと比較して、組織の全体的な競争力に影響を与えていることに同意しています。」 開発サイクルが短いほど、組織が市場time-to-marketの目標を達成し、適切なタイミングで適切な機会を活用できる可能性が高くなります。
- 速度 – 通常、完全な CI/CD パイプラインは数時間以内にソフトウェアの変更を顧客にリリースできます。特に、迅速な障害分離や小さなパッチプッシュの場合、CI/CD パイプラインは平均復旧時間 (MTTR) を改善するのに役立ちます。詳細については、「MTTR の削減」を参照してください。
- セキュリティ — 完全な CI/CD パイプラインは、攻撃のエントリポイントの可能性を減らし、人為的ミスのリスクを減らすことで、リリースプロセスも保護します。完全に自動化された CI/CD パイプラインによるセキュリティ上のメリットは、データ侵害、サービス停止などによるコストのかかる結果を回避するのに役立ちます。
- 削減率の低下 – 開発者は、優れた機能の作成により多くの時間を費やし、メンテナンスとデバッグの無限サイクルで時間を短縮できると、より満足します。組織にとって、これはトップの人材を長期間獲得して維持することを意味します。
- 「品質評価コードの提供」 — 開発者はコードを小さなバッチで共有リポジトリにリリースします。これにより、パラレルテストを実行できます (BrowserStack ブログ記事) 。単独で作業するのではなく、ビルドをチームと頻繁に共有し、共同で重要なバグを特定します。これにより、デベロッパーがサポートされるため、不正なコードが本番稼働用になるのを防ぐことができます。デベロッパーピアからのサポートは、高品質のリリースに貢献し、組織の成長を促進します。
- メンテナンス – メンテナンスと更新は、優れた製品を作る上で重要な部分です。ただし、ピークトラフィック時にはシステムを停止しないでください。CI/CD パイプラインを使用すると、使用頻度の低い時間帯にメンテナンスを実行し、ダウンタイムとパフォーマンスへの影響を最小限に抑えることができます。

CI/CD について

継続的インテグレーションと継続的デリバリー (CI/CD) は、ソフトウェアリリースのライフサイクルを自動化するプロセスです。場合によっては、CI/CD の D がデプロイを意味することもあります。継続的デリバリーと継続的デプロイの違いは、本番環境に変更をリリースするときに発生します。継続的デリバリーでは、本番環境に変更を昇格させる前に、手動で承認する必要があります。継続的デプロイでは、パイプライン全体を通じて中断のないフローが使用され、明示的な承認は必要ありません。この戦略では一般的な CI/CD の概念について説明しているため、提供される推奨事項と情報は、継続的デリバリーと継続的デプロイの両方のアプローチに適用されます。

CI/CD は、コミットから本番環境に新しいコードを取得するために従来必要とされていた手動プロセスの多くまたはすべてを自動化します。CI/CD パイプラインには、ソース、ビルド、テスト、ステージング、本番ステージが含まれます。各ステージで、CI/CD パイプラインはコードをデプロイまたはテストするために必要なインフラストラクチャをプロビジョニングします。CI/CD パイプラインを使用することで、開発チームはコードを変更し、自動的にテストしてデプロイにプッシュできます。

基本的な CI/CD プロセスを確認してから、意図的または無意識に、完全な CI/CD から逸脱する方法をいくつか検討しましょう。次の図は、各ステージの CI/CD ステージとアクティビティを示しています。



継続的インテグレーションについて

継続的な統合は、 の Git リポジトリなどのコードリポジトリで行われますGitHub。1 つのメインブランチをコードベースの信頼できるソースとして扱い、機能開発用の存続期間の短いブランチを作成します。機能を上位環境にデプロイ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。機能ブランチが上位環境に直接デプロイされることはありません。詳細については、このガイドの「トランクベースのアプローチ」を参照してください。

継続的な統合プロセス

1. 開発者は、メインブランチから新しいブランチを作成します。
2. 開発者は変更を行い、ローカルでビルドとテストを行います。
3. 変更の準備ができたら、開発者はメインブランチを送信先とするプルリクエスト (GitHub ドキュメント) を作成します。
4. コードがレビューされます。
5. コードが承認されると、メインブランチにマージされます。

継続的デリバリーについて

継続的デリバリーは、開発環境や本番環境など、隔離された環境で行われます。各環境で発生するアクションは異なる場合があります。多くの場合、最初のステージの 1 つを使用してパイプライン自体を更新してから続行します。デプロイの最終結果は、各環境が最新の変更で更新されることです。構築とテスト用の開発環境の数も異なりますが、少なくとも 2 つを使用することをお勧めします。パイプラインでは、各環境は重要度の高い順に更新され、最も重要な環境である本番環境で終わります。

継続的な配信プロセス

パイプラインの継続的配信部分は、ソースリポジトリのメインブランチからコードをプルし、ビルドステージに渡すことで開始されます。リポジトリの Infrastructure as Code (IaC) ドキュメントは、各ステージで実行されるタスクの概要を示しています。IaC ドキュメントの使用は必須ではありませんが、 AWS CloudFormation や などの IaC サービスまたはツールAWS Cloud Development Kit (AWS CDK)を使用することを強くお勧めします。最も一般的な手順は次のとおりです。

1. ユニットテスト
2. コードビルド

3. リソースプロビジョニング
4. 統合テスト

エラーが発生した場合、またはパイプラインの任意の段階でテストが失敗した場合、現在のステージは以前の状態にロールバックされ、パイプラインは終了します。それ以降の変更はコードリポジトリで開始し、完全な CI/CD プロセスを実行する必要があります。

CI/CD パイプラインのテスト

デプロイパイプラインで一般的に参照される 2 種類の自動テストは、ユニットテストと統合テストです。ただし、コードベースと開発環境で実行できるテストには多くのタイプがあります。AWS デプロイパイプラインリファレンスアーキテクチャでは、次のタイプのテストを定義します。

- ユニットテスト – これらのテストでは、アプリケーションコードを構築して実行し、期待どおりに動作していることを確認します。コードベースで使用されるすべての外部依存関係をシミュレートします。ユニットテストツールの例には、JUnit、Jest、pytest などがあります。
- 統合テスト – これらのテストでは、プロビジョニングされたテスト環境に対してテストすることで、アプリケーションが技術要件を満たしていることを検証します。統合テストツールの例には、Cucumber、vRest NG、integ-tests (の場合) などがあります AWS CDK。
- 承認テスト – これらのテストでは、プロビジョニングされたテスト環境に対してテストすることで、アプリケーションがユーザー要件を満たしていることを確認します。受け入れテストツールの例としては、Cypress や Selenium などがあります。
- 合成テスト – これらのテストはバックグラウンドで継続的に実行され、トラフィックを生成してシステムが正常であることを確認します。合成テストツールの例としては、Amazon CloudWatch Synthetics や Dynatrace Synthetic Monitoring などがあります。
- パフォーマンステスト – これらのテストは、本番稼働容量をシミュレートします。アプリケーションがパフォーマンス要件を満たしているかどうかを判断し、メトリクスを過去のパフォーマンスと比較します。パフォーマンステストツールの例としては、Apache JMeter、Locust、Gatling などがあります。
- 耐障害性テスト – カオステストとも呼ばれ、これらのテストでは、リスク領域を特定するために環境に障害を発生させます。その後、障害が挿入された期間は、障害がない期間と比較されます。耐障害性テストツールの例としては、 AWS Fault Injection Serviceや Gremlin などがあります。
- 静的アプリケーションセキュリティテスト (SAST) – これらのテストでは、SQL インジェクションやクロスサイトスクリプティング (XSS) などのセキュリティ違反がないかコードを分析します。SAST ツールの例には、Amazon CodeGuru、SonarQube、Checkmarx などがあります。

- 動的アプリケーションセキュリティテスト (DAST) – これらのテストは、ペンテストまたはペンテストとも呼ばれます。これらは、プロビジョニングされたテスト環境で SQL インジェクションや XSS などの脆弱性を識別します。DAST ツールの例には、Zed Attack Proxy (ZAP) や HCL AppScan などがあります。詳細については、「ペネトレーションテスト」を参照してください。

すべての CI/CD パイプラインがこれらのテストをすべて実行するわけではありません。ただし、少なくとも、パイプラインはコードベースでユニットテストと SAST テストを実行し、テスト環境で統合テストと承認テストを実行する必要があります。

CI/CD パイプラインのメトリクス

AWS デプロイパイプラインリファレンスアーキテクチャに従って、少なくとも CI/CD パイプラインの次の 4 つのメトリクスを追跡する必要があります。

- リードタイム – 1 回のコミットが本番環境に到達するまでにかかる平均時間。ユースケースに応じて、リードタイムを 1 時間から 1 日の間で設定することをお勧めします。
- デプロイ頻度 – 一定期間内の本番稼働用デプロイの数。ユースケースに応じて、デプロイ頻度を 1 日に複数回から 1 週間に 2 回に絞ることをお勧めします。
- 平均障害間隔 (MTBF) — 成功したパイプラインの開始から失敗したパイプラインの開始までの平均時間。できるだけ高い MTBF をターゲットにすることをお勧めします。詳細については、「MTBF の増加」を参照してください。
- 平均復旧時間 (MTTR) — 失敗したパイプラインの開始から次の成功したパイプラインの開始までの平均時間。できるだけ低い MTTR をターゲットにすることをお勧めします。詳細については、「MTTR の削減」を参照してください。

これらのメトリクスは、チームが完全に CI/CD になるための進捗状況を追跡するのに役立ちます。チームは、最適な目標について、組織のステークホルダーとオープンに話し合う必要があります。状況とニーズは、組織ごとに、またチームごとに大きく異なります。

急激な変化は、通常、問題が発生するリスクを高めることを覚えておくことが重要です。小規模で段階的な改善を目的とした目標を設定します。完全な CI/CD パイプラインの一般的な最適なリードタイムは 3 時間未満です。リードタイムが 5.2 日で始まるチームは、数週間に 1 日の削減を目標とする必要があります。このチームがリードタイムを 1 日以下にすると、チームや組織の関係者が必要と判断した場合のみ、数か月間そこに留まり、より積極的なリードタイムに移行できます。

CI/CD プロセスの違い

CI/CD パイプラインは、最新のトランクベースのワークフローを使用します。このワークフローでは、デベロッパーは、CI/CD パイプラインの CD 部分を通じて構築およびテストされるメインブランチ (またはトランク) に、小規模で頻繁な更新をマージします。このワークフローは、開発ブランチとリリースブランチをリリーススケジュールで区切る Gitflow ワークフローを置き換えました。多くの組織では、Gitflow はまだバージョン管理とデプロイの一般的な方法です。ただし、現在はレガシーと見なされており、CI/CD パイプラインに統合するのは難しい場合があります。

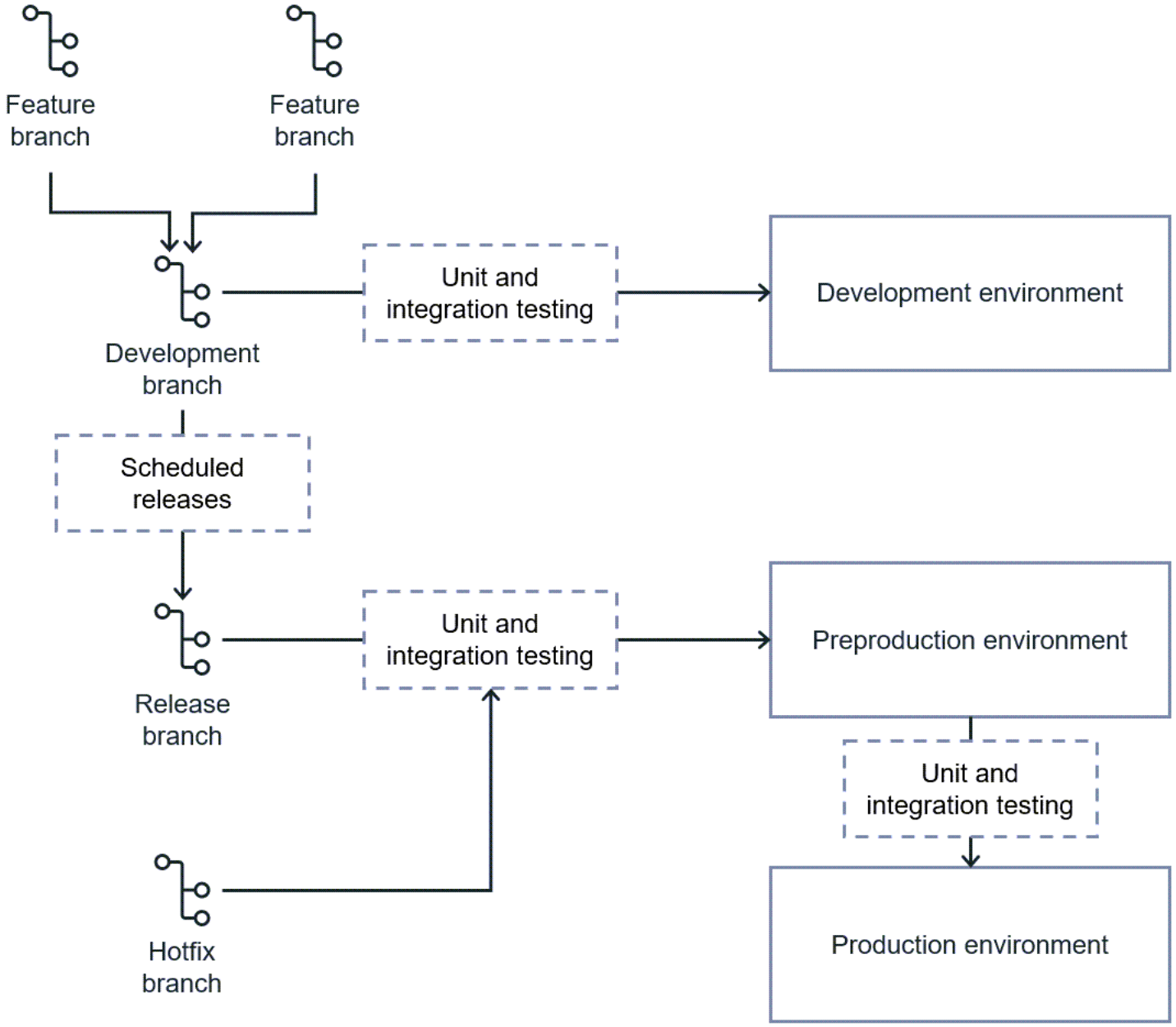
多くの組織では、Gitflow ワークフローからトランクベースのワークフローへの移行は不完全であり、その結果、途中でどこかにスタックし、CI/CD に完全に移行することはありません。ある意味、パイプラインはレガシーワークフローの特定の残余物に縛られ、過去と現在の間の移行状態でスタックします。Git ワークフローの違いを確認し、レガシーワークフローの使用が以下にどのように影響するかを学習します。

- 環境の整合性
- リリース
- セキュリティ

最新の設定でレガシー Git ワークフローの残余を簡単に識別できるように、Gitflow を最新のトランクベースのアプローチと比較してみましょう。

Gitflow アプローチ

次の図は、Gitflow ワークフローを示しています。Gitflow アプローチでは、複数のブランチを使用して、複数の異なるバージョンのコードを同時に追跡します。デベロッパーが最新バージョンのコードで作業している間、将来のいずれかの時点でアプリケーションの更新のリリースをスケジュールします。トランクベースのリポジトリでは、機能フラグを使用してこれを実現できますが、デフォルトで Gitflow に組み込まれています。

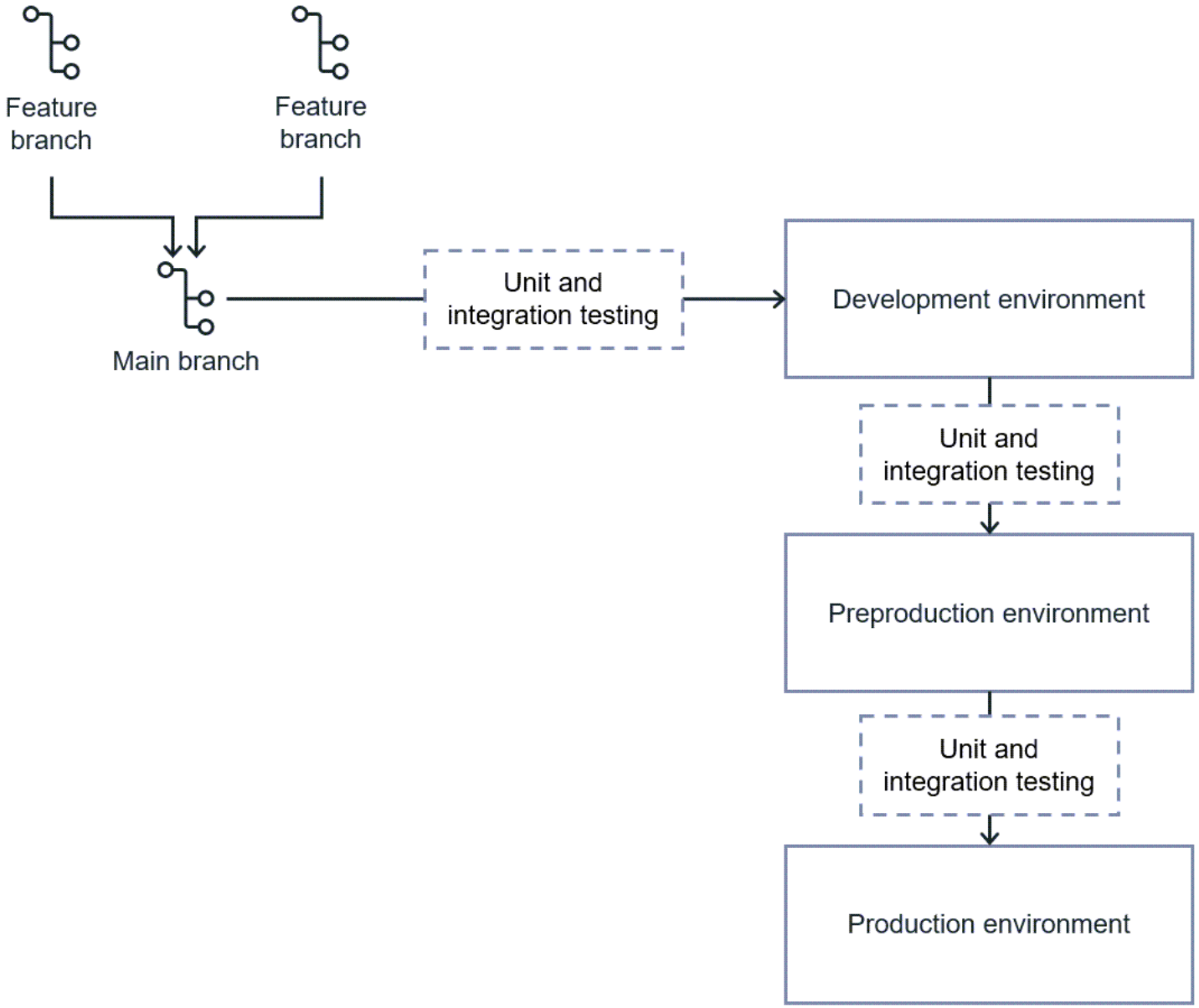


Gitflow アプローチの結果の 1 つは、アプリケーション環境が通常同期していないことです。標準の Gitflow 実装では、開発環境はコードの現在の状態を反映しますが、本番稼働前環境と本番稼働環境は最新のリリースのコードベースの状態のままです。

これにより、デベロッパーが作業するコードベースをリリースされていない機能を公開せずに本番環境にマージできないため、本番環境に不具合が発生した場合に、作業が複雑になります。Gitflow がこの状況を処理する方法は、修正プログラムを使用することです。ホットフィックスブランチはリリースブランチから作成され、上位環境に直接デプロイされます。その後、コードを最新の状態に保つために、修正ブランチが開発ブランチにマージされます。

トランクベースのアプローチ

次の図は、トランクベースのワークフローを示しています。トランクベースのワークフローでは、デベロッパーは特徴量ブランチでローカルに特徴量を構築してテストし、それらの変更をメインブランチにマージします。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。ユニットテストと統合テストは、各環境間で行われます。



このワークフローを使用すると、すべての環境が同じコードベースで動作します。リリースされていない機能を公開せずにメインブランチに変更を実装できるため、上位環境のホットフィックスブランチは必要ありません。メインブランチは常に安定しており、欠陥がなく、解放できる状態であると想

定されます。これにより、CI/CD パイプラインのソースとして統合し、パイプライン内のすべての環境を通じてコードベースを自動的にテストしてデプロイできます。

トランクベースのアプローチによる環境整合性のメリット

多くのデベロッパーが知っているように、コードの 1 つの変更によってバタフライ効果が生じることがあります (アメリカのサイエンティストの記事) 。ここでは、無関係に見えるわずかな偏差によって、予期しない結果を引き起こす連鎖反応が始まることがあります。その後、開発者は根本原因を発見するために完全に調査する必要があります。

科学者が実験を行うと、テスト対象を実験グループとコントロールグループの 2 つのグループに分けます。目的は、実験グループとコントロールグループを、実験でテストされているものを除いて完全に同一にすることです。実験グループで、コントロールグループで発生しない何かが発生した場合、唯一の原因はテスト対象のモノです。

デプロイの変更を実験グループとして考え、各環境を個別のコントロールグループとして考えます。より低い環境でのテストの結果は、コントロールが上位環境と同じ場合にのみ信頼性があります。環境が逸脱するほど、上位環境の欠陥を検出する可能性が高くなります。つまり、コードの変更が本番環境で失敗する場合は、最初にベータ版で失敗して、本番環境に移行しないようにすることをお勧めします。そのため、テスト環境が最も低い環境から本番稼働環境まで、各環境を同期させるようにあらゆる努力をする必要があります。これは環境の整合性と呼ばれます。

完全な CI/CD プロセスの目標は、できるだけ早く問題を発見することです。トランクベースのアプローチを使用して環境の整合性を維持することで、修正の必要性を実質的に排除できます。トランクベースのワークフローでは、本番環境で問題が最初に表示されることはまれです。

Gitflow アプローチでは、修正プログラムが上位環境に直接デプロイされた後、開発ブランチに追加されます。これにより、今後のリリースの修正が保持されます。ただし、ホットフィックスはアプリケーションの現在の状態から直接開発され、テストされました。ホットフィックスが本番環境で完全に機能する場合でも、開発ブランチの新しい機能とやり取りするときに問題が発生する可能性があります。通常、ホットフィックスにホットフィックスをデプロイすることは望ましくないため、開発者はホットフィックスを開発環境に改良しようとして余分な時間を費やすことになります。多くの場合、これは重大な技術的負債につながり、開発環境の全体的な安定性を低下させる可能性があります。

環境で障害が発生すると、すべての変更がロールバックされ、環境が以前の状態に戻ります。コードベースを変更すると、最初のステージからパイプラインが再び開始されます。本番環境で問題が発生した場合は、パイプライン全体も修正する必要があります。通常、より低い環境を通過するのにかか

る余分な時間は、このアプローチを使用して回避される問題と比較してごくわずかです。下位環境の目的は、本番環境に到達する前にミスをキャッチすることであるため、Gitflow アプローチでこれらの環境をバイパスすることは非効率で不要なリスクです。

トランクベースのアプローチの利点をリリースする

多くの場合、修正が必要な問題の 1 つは、レガシーワークフローでは、デベロッパーが作業しているアプリケーションの状態には、本番環境にまだ存在しない未リリースの機能がいくつか含まれている可能性があることです。本番稼働環境と開発環境は、スケジュールされたリリースが発生したときにのみ同期され、次にスケジュールされたリリースまですぐに再び分岐し始めます。

スケジュールされたリリースは、完全な CI/CD プロセス内で行うことができます。機能フラグを使用すると、コードの本番稼働へのリリースを遅らせることができます。ただし、完全な CI/CD プロセスにより、スケジュールされたリリースが不要になり、柔軟性が高まります。結局のところ、連続は CI/CD のキーワードであり、変更の準備ができたらリリースされることを示します。ほとんどの場合、下位のテスト環境と同期しない別のリリース環境を維持しないでください。

パイプラインが完全に CI/CD でない場合、通常、上位環境と下位環境の相違はブランチレベルで発生します。デベロッパーは開発ブランチで作業し、スケジュールされたリリースの時間帯にのみ更新される別のリリースブランチを維持します。リリースブランチと開発ブランチが分岐すると、他の複雑さが発生する可能性があります。

環境が同期していないだけでなく、デベロッパーが開発ブランチで作業し、本番環境よりもはるかに進んでいるアプリケーション状態に慣れるにつれて、問題が発生するたびに本番環境に再調整する必要があります。開発ブランチの状態は、本番稼働前に多くの機能になる可能性があります。デベロッパーが毎日そのブランチで作業する場合、本番環境にリリースされているものとリリースされていないものを覚えておくのは困難です。これにより、他のバグを修正するプロセス中に新しいバグが発生するリスクが高まります。その結果、タイムラインを延長し、機能リリースを数週間、数か月、さらには数年間遅らせる修正の無限のサイクルのように見えます。

トランクベースのアプローチによるセキュリティ上の利点

CI/CD プロセスにより、デプロイに対する完全な自動化の単一ソースの真のアプローチが提供されます。パイプラインには単一のエントリポイントがあります。ソフトウェア更新は、最初からパイプラインに入り、環境間でそのまま渡されます。パイプラインの任意の段階で問題が検出された場合、修正するコードは同じプロセスを通過し、最初のステージで開始する必要があります。パイプラインのエントリポイントを減らすことで、パイプラインに脆弱性が発生する可能性も低くなります。

さらに、エントリポイントは本番環境から可能な限り遠いポイントであるため、脆弱性が本番環境に到達する可能性が大幅に低下します。完全な CI/CD パイプラインに手動承認プロセスを実装しても、変更を次の環境に昇格させるかどうかについて、Go または No-Go の意思決定を許可できます。意思決定者は、必ずしも変更をデプロイするのと同じ人物ではありません。これにより、コード変更のデプロイ者とそれらの変更の承認者の責任が分離されます。また、技術の少ない組織のリーダーが承認者の役割を果たすことも可能になります。

最後に、単一のエントリポイントを使用すると、本番環境のユーザーインターフェイス (UI) コンソールへの書き込みアクセスを数人またはゼロユーザーに制限できます。コンソールで手動で変更できるユーザーの数を減らすことで、セキュリティイベントのリスクを軽減できます。本番環境でコンソールを手動で管理する機能は、CI/CD 自動アプローチよりもレガシーワークフローではるかに必要です。これらの手動の変更の追跡、レビュー、テストはより困難です。通常、時間を節約するために実行されますが、長期的にはプロジェクトに多大な技術的負担がかかります。

コンソールのセキュリティ問題は、必ずしも悪意のある人物によって引き起こされるわけではありません。コンソールで発生する問題の多くは偶発的です。偶発的なセキュリティへの露出は非常に一般的であり、ゼロトラストセキュリティモデルが台頭しています。このモデルでは、内部スタッフでも最小特権のアクセス許可とも呼ばれるアクセス許可ができるだけ少ない場合、セキュリティ上の事故の可能性が低くなります。すべてのプロセスを自動パイプラインに制限することで、本番環境の整合性を維持することで、コンソール関連のセキュリティ問題のリスクを実質的に排除できます。

CI/CD パイプラインのリトムテスト

化学では、リトマス紙は特殊な赤または青の色合いで処理された薄い紙片で、物質の変性を判断するために使用されます。アジットは青リトムス紙を赤に、ベースは赤リトムス紙を青に、中立的な物質は紙の色にまったく影響しません。

リトム紙がアルコール度を決定する方法は、物質のリンレベルを測定することです。レベルが 8 より大きい場合、それは古く、5 より小さい場合は基本、5 から 8 の間であれば中立です。同様に、CI/CD リトムテストは、パイプラインの CI/CD レベルを測定するのに役立ちます。

パイプラインが完全に CI/CD であるかどうかをテストするには

1. 0 のスコアから始めます。
2. 次の各質問に回答し、はいと答えるたびにスコアに 1 を追加します。
    - リポジトリにはそれぞれ、環境にデプロイするために使用されるメインブランチが 1 つだけありますか？
    - メインブランチに頻繁にコードをコミットし、長時間実行される特徴量ブランチは避けていますか？
    - パイプラインには単一のエントリポイントがありますか？ つまり、パイプラインは各リポジトリからコードを 1 回だけプルしますか？
    - デプロイ環境が複数ありますか？
    - パイプラインが実行されていない場合、上位環境と下位環境は一般的に同期されていますか？
    - デプロイする前にコードでテストを実行しますか？
    - 次の環境に昇格する前に、環境でテストを実行しますか？
    - パイプラインは完全なロールバックを行い、障害後に終了しますか？
    - 障害から復旧するときに、パイプラインは最初のステップから再開されますか？
    - 本番環境のバグを修正するには、機能を本番環境にリリースする場合と同じプロセスに従いますか？
    - コードをデプロイするために何らかの形式のInfrastructure as Code (IaC) テンプレートを使用していますか？
3. 以下の各質問に回答し、いいえと答えるたびにスコアに 1 を追加します。
    - メインブランチ以外のブランチからデプロイ環境に直接デプロイしたことはありますか？
    - ブランチから上位環境または本番環境に直接デプロイしたことはありますか？

- 多くの場合、上位の環境ではバグが発見され、下位の環境では発見されないことがありますか？
- デプロイ中に低い環境をバイパスしたことはありますか？
- スケジュールされたリリース時間が本番環境にデプロイされるまで待ちますか？
- 本番環境のコンソールで定期的に更新していますか？
- デプロイを完了するには、本番環境のコンソールで手動デプロイ手順を実行する必要がありますか？
- 複数のユーザーが本番環境への書き込みアクセス権を持っていますか？
- 5 人以上のユーザーが本番環境への書き込みアクセス権を持っていますか？

4. スコアを 2 で割ります。これはパイプラインの CI/CD スコアです。
5. パイプラインの CI/CD スコアを次の表と比較して、パイプラインの CI/CD レベルを判断します。

CI/CD スコア	CI/CD レベル
9.5 以上	完全な CI/CD
8 ~ 9	ほとんど CI/CD
5 ~ 7	[Neutral] (ニュートラル)
5 未満	CI/CD ではない

スコアが 8 未満の場合は、次のレベルに徐々に移行する目標を設定することをお勧めします。その目標が達成されたら、製品の利害関係者は、新しい目標を設定するかどうか、いつ設定すべきかを評価する必要があります。この演習の目的は、必ずしもパイプラインへの変更を主張するのではなく、完全な CI/CD デプロイプロセスがどのように見えるか、パイプラインが現在その領域にある場所を把握することです。

CI/CD パイプラインのベストプラクティス

完全な CI/CD パイプラインのベストプラクティスを次に示します。

- 本番環境の保護 – IaC を使用してアカウントと環境のメンテナンスに必要な事実上すべてを達成できるため、コンソールとプログラムによるアクセスを制限して、本番環境を保護するためにあらゆる努力をすることが重要です。アクセスを少数またはゼロのユーザーのみに制限することをお勧めします。を通じて IaC をデプロイする場合 AWS CloudFormation、ユーザーには制限されたアクセス許可が必要です。ほとんどのアクセス許可は、サービスロールを介して CloudFormation サービスに割り当てられます。詳細については、CloudFormation ドキュメントの「サービスロール」および「 の最小特権アクセス許可のポリシーの実装 AWS CloudFormation」を参照してください。
- 環境ごとに個別のアカウントを作成する – 環境ごとに個別のアカウントを割り当てることで、デプロイプロセスを簡素化し、アカウントレベルできめ細かなアクセスコントロールを作成できます。複数の環境がリソースを共有すると、環境の整合性が分離されたユニットとして低下します。環境を同期させ、区別しておくことをお勧めします。これは、そのアカウントのすべてが本番稼働用リソースとして扱われる必要があるため、本番環境にとってさらに重要です。
- 個人を特定できる情報 (PII) を本番環境に制限する – どちらも、責任リスクに対するセキュリティと保護の両方を目的として、PII を可能な限り保護します。より低い環境で可能な場合は、本番環境から機密データをコピーする代わりに、匿名化されたデータまたはサンプルデータを使用します。
- リポジトリ内のコードを確認する – 完全な CI/CD プロセスにより、パイプラインのエントリポイントが 1 つのポイントに減り、その 1 つのポイントを保護する必要があります。このため、機能ブランチをメインブランチにマージする前に、複数のコードレビューが必要です。これらのコードレビューは、資格のあるチームメンバーによって実施できますが、少なくとも 1 人の上級メンバーがレビューする必要があります。コードはレビューワーによって厳密にテストする必要があります。パイプラインの問題を修正する最善の方法は、パイプラインに問題が取り込まれないようにすることです。また、マージする前に、レビューワーが行ったすべてのコメントを解決することが重要です。この解決策は、単に変更が必要ない理由を説明することかもしれませんが、すべてのコメントに対処することは、パイプラインに問題の導入を防ぐために重要な追加チェックです。
- 小規模で頻繁なマージを行う – 継続的な統合を最大限に活用するには、ローカルの変更をパイプラインに継続的にプッシュすることをお勧めします。結局のところ、ローカル環境もそれに追いついていれば、開発環境が同期し続ける方がはるかに有益です。

CI/CD パイプラインのベストプラクティスの詳細については、「継続的な統合と継続的デリバリーの実践 AWS」の「ベストプラクティスの概要」を参照してください。

よくある質問

デプロイプロセスが完全に CI/CD ではないことを示す重要な指標は何ですか？

最も一般的な指標は、パイプライン内の個別の環境を表す複数のリポジトリブランチがある場合です。完全な CI/CD プロセスのリポジトリは、トランクベースのワークフローを使用します。このワークフローでは、1 つのブランチがそのリポジトリのデプロイの唯一の信頼できるソースとして機能します。詳細については、「トランクベースのアプローチ」を参照してください。その他の指標には、シンプルな実行または実行禁止の決定、修正プログラムの使用、スケジュールされたリリース以外の手動デプロイステップが含まれます。

完全な CI/CD プロセスを使用しても、特定の時点における特定の機能のリリースをスケジュールしたい場合はどうなりますか？

これは通常、機能フラグを使用して行われます。このプロセスでは、デプロイは引き続き行われますが、特定の機能は、リリースするまでコードで条件付きクロージャを使用して非表示になります。

デプロイプロセスの一部のステップを自動化できない場合はどうなりますか？

完全な CI/CD パイプラインの目標の 1 つは、手動プロセスの必要性を最小限に抑えることですが、手動プロセスが必要になる可能性のあるユースケースは確実にあります。実際、コンサルティングアプリケーションログなどの読み取り専用プロセスは、リスクを最小限に抑えながら本番環境で実行できます。ただし、本番環境での手動書き込みアクションは絶対的な最後の手段として扱うことを強くお勧めします。

技術スタッフが、完全な CI/CD プロセスよりもレガシーワークフローに慣れている場合はどうなりますか？

技術スタッフが大きな変更に耐性を持つことは一般的です。特に、ベストプラクティスであったものが新しいものに置き換えられた場合です。テクノロジーは急速に動き、改善が絶えず発見されています。ある程度の懐疑論は技術スタッフにとっては良い品質ですが、変化を受け入れることも重要で

す。システムへの変更を実装する前に管理する必要があるため、懐疑的なスタッフによる移動が速すぎないようにしてください。重要なのは、懐疑論者が永久に静的のままにならないようにすることです。

環境が複数のアカウントにある場合はどうなりますか？ それでも完全な CI/CD プロセスを使用できますか？

はい。実際には、環境ごとに個別のアカウントを使用することをお勧めします。異なるアカウントでステージをアクティブ化するパイプラインの詳細については、「Create a pipeline in CodePipeline that uses resources from another AWS アカウント」を参照してください。

次のステップ

CI/CD パイプラインのリトムテスト セクションを使用して、組織内の DevOps プロセスを評価します。プロセスが完全に CI/CD であるかどうかを判断します。そうでない場合は、CI/CD デプロイの利点を最大限に活用するために改善が必要かどうかを判断します。

いつ完了したかはどうすればわかりますか？ 答えは、多くの組織が実際には終了しないということです。途中で、ユースケースに適した場所に停止します。完全な CI/CD パイプラインは最適なシナリオですが、組織の状況と決定の背後にあるステークホルダーに大きく依存します。ステークホルダーは、CI/CD 実装のどの段階がユースケースに最適で、進行を次の段階にどのようにマッピングするのが最適かを決定する必要があります。

CI/CD パイプラインの設計と構築の詳細については、「」を参照してくださいリソース。

リソース

AWS ドキュメントとリファレンス

- AWS Deployment Pipeline Reference Architecture
- 継続的デリバリーとは
- での継続的インテグレーションと継続的デリバリーの実践 AWS (AWS ホワイトペーパー)
- で CI/CD パイプラインを設定する AWS (AWS 実践的なチュートリアル)
- をサポート AWS リージョン していないパイプラインを に AWS CodePipeline作成する (AWS 規範ガイダンス)
- Deployment Pipelines リファレンスアーキテクチャとリファレンス実装 (AWS ブログ記事)

サービスとツール

- CI/CD リトムテスト
- AWS Cloud Development Kit (AWS CDK)
- AWS CloudFormation
- AWS CodePipeline

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	—	2023 年 8 月 25 日

AWS 規範ガイダンス用語集

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

- 使用停止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

ABAC

属性ベースのアクセスコントロールを参照してください。

抽象化されたサービス

「マネージドサービス」を参照してください。

ACID

アトミック性、一貫性、分離性、耐久性を参照してください。

アクティブ - アクティブ移行

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

アクティブ - パッシブ移行

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

集計関数

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

AI

「人工知能」を参照してください。

AIOps

「人工知能オペレーション」を参照してください。

匿名化

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

アンチパターン

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

アプリケーションコントロール

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

アプリケーションポートフォリオ

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

人工知能 (AI)

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

AI オペレーション (AIOps)

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

非対称暗号化

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

原子性、一貫性、分離性、耐久性 (ACID)

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

属性ベースのアクセス制御 (ABAC)

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

信頼できるデータソース

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

アベイラビリティーゾーン

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

AWS クラウド導入フレームワーク (AWS CAF)

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は、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 るのに役立ちます。 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

個人や組織を混乱させたり、損害を与えたりすることを意図したボット。

BCP

「事業継続計画」を参照してください。

動作グラフ

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

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。エンディアン性も参照してください。

二項分類

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

ブルームフィルター

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

ブルー/グリーンデプロイ

2 つの異なる同一の環境を作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (青) で実行し、新しいアプリケーションバージョンを別の環境 (緑) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティや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、ボットハーダーまたはボットオペレーターとして知られる、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。ボットは、ボットとその影響をスケールするための最もよく知られているメカニズムです。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発したり、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(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なります。詳細については、 Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータ AWS 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、ホワイトペーパー AWSでのコンテナ化されたマイクロサービスの実行 の ビジネス機能を中心に組織化 セクションを参照してください。

ビジネス継続性計画 (BCP)

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

C

CAF

AWS 「クラウド導入フレームワーク」を参照してください。

Canary デプロイ

エンドユーザーへのバージョンのスローリリースと増分リリース。確信が持てば、新しいバージョンをデプロイし、現在のバージョン全体を置き換えます。

CCoE

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

CDC

「データキャプチャの変更」を参照してください。

変更データキャプチャ (CDC)

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

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストします。AWS Fault Injection Service （ AWS FIS ） を使用して、 AWS ワークロードにストレスを与え、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

ターゲットがデータ AWS のサービス を受信する前のローカルでのデータの暗号化。

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

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

クラウド運用モデル

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

導入のクラウドステージ

組織が に移行するときに通常実行する 4 つのフェーズ AWS クラウド :

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

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略と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

「設定管理データベース」を参照してください。

コードリポジトリ

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

コールドキャッシュ

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

コールドデータ

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

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオなどのビジュアル形式から情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI は CV 用の画像処理アルゴリズムを提供します。

設定ドリフト

ワークロードの場合、設定は想定状態から変化します。ワークロードが非準拠になる可能性があり、通常は段階的かつ意図的ではありません。

構成管理データベース (CMDB)

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

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョン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

一元管理とガバナンスを備えた分散型の分散型データ所有権を提供するアーキテクチャフレームワーク。

データ最小化

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

データ境界

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

データの前処理

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

データ出所

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

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには通常、大量の履歴データが含まれており、通常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る。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

ディープラーニング

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

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。

デプロイ

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

開発環境

「環境」を参照してください。

検出管理

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

開発バリューストリームマッピング (DVSM)

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

デジタルツイン

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

ディメンションテーブル

スタースキーマでは、ファクトテーブル内の量的データに関するデータ属性を含む小さなテーブル。ディメンションテーブル属性は通常、テキストフィールドまたはテキストのように動作する

離散数値です。これらの属性は、クエリの制約、フィルタリング、結果セットのラベル付けに一般的に使用されます。

ディザスタ

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

ディザスタリカバリ (DR)

災害によるダウンタイムとデータ損失を最小限に抑えるために使用する戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

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

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。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 、ガバナンス要件のコンプライアンスに影響を与える可能性のあるランディングゾーンの変更を検出したりできます。

DVSM

「開発値ストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

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、応答時間を短縮できます。

電子データ交換 (EDI)

組織間の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プロセス。

暗号化キー

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

エンディアン

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

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これら

のアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (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。たとえば、 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

星スキーマの中央テーブル。事業運営に関する量的データを保存します。通常、ファクトテーブルには、メジャーを含む列とディメンションテーブルへの外部キーを含む列の 2 つのタイプの列が含まれます。

フェイルファスト

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

障害分離の境界

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

同様のタスクの実行を求める前に、タスクと必要な出力を示す少数の例を LLM に提供します。この手法は、プロンプトに埋め込まれた例 (ショット) からモデルが学習するコンテキスト内学習のアプリケーションです。少数ショットプロンプトは、特定のフォーマット、推論、またはドメインの知識を必要とする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コントロール」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

段階的なアプローチを使用する代わりに、変更データキャプチャによる継続的なデータレプリケーションを使用して、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

一般化データとラベル付けされていないデータの大規模なデータセットでトレーニングされている大規模な深層学習ニューラルネットワーク。FMs は、言語の理解、テキストと画像の生成、自然言語の会話など、さまざまな一般的なタスクを実行できます。詳細については、「基盤モデルとは」を参照してください。

G

生成 AI

大量のデータでトレーニングされ、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る AI モデルのサブセット。詳細については、「生成 AI とは」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 ワークフローはレガシーと見なされ、トランクベースのワークフローはモダンで推奨されるアプローチです。

ゴールデンイメージ

そのシステムまたはソフトウェアの新しいインスタンスをデプロイするためのテンプレートとして使用されるシステムまたはソフトウェアのスナップショット。例えば、製造では、ゴールデンイメージを使用して複数のデバイスにソフトウェアをプロビジョニングし、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、、Amazon GuardDuty AWS Security Hub、、 AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

ハイアベイラビリティ (HA)

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

ヒストリアンのモダナイゼーション

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

ホールドアウトデータ

機械学習モデルのトレーニングに使用されるデータセットから保留される、ラベル付きの履歴データの一部。モデル予測をホールドアウトデータと比較することで、ホールドアウトデータを使用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

「産業用モノのインターネット」を参照してください。

イミュータブルインフラストラクチャ

既存のインフラストラクチャを更新、パッチ適用、または変更する代わりに、本番環境のワークロード用に新しいインフラストラクチャをデプロイするモデル。イミュータブルインフラストラクチャは、本質的にミュータブルインフラストラクチャよりも一貫性、信頼性、予測性が高くなります。詳細については、 AWS 「 Well-Architected Framework」の「Deploy using immutable infrastructure best practice」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 によって導入された用語で、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ゼーションを指します。

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。

産業分野における IoT (IIoT)

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

インスペクション VPC

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

IoT

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

解釈可能性

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

IoT

「モノのインターネット」を参照してください。

IT 情報ライブラリ (ITIL)

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

大量のデータに対して事前トレーニングされた深層学習 AI モデル。LLM は、質問への回答、ドキュメントの要約、テキストの他の言語への翻訳、文の完了など、複数のタスクを実行できます。詳細については、LLMs」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースのアクセスコントロール」を参照してください。

最小特権

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

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。エンディアン性も参照してください。

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またはプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスにつながる可能性があります。マルウェアの例としては、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス 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。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、マネージドサービスの例です。これらは抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するためのソフトウェアシステムで、原材料を工場の完成製品に変換します。

MAP

「移行促進プログラム」を参照してください。

メカニズム

ツールを作成し、ツールの導入を推進し、調整を行うために結果を検査する完全なプロセス。メカニズムは、動作時にそれ自体を強化および改善するサイクルです。詳細については、 AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

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

MES

「製造実行システム」を参照してください。

メッセージキューイングテレメトリトランスポート (MQTT)

リソースに制約のある IoT デバイス用の、パブリッシュ/サブスクライブパターンに基づく軽量 machine-to-machine (M2M) 通信プロトコル。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

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

Migration Acceleration Program (MAP)

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)

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

移行準備状況評価 (MRA)

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

移行戦略

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

ML

???「機械学習」を参照してください。

モダナイゼーション

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

モダナイゼーション準備状況評価

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

モノリシックアプリケーション (モノリス)

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

MPA

「移行ポートフォリオ評価」を参照してください。

MQTT

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

多クラス分類

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

ミュータブルインフラストラクチャ

本番ワークロードの既存のインフラストラクチャを更新および変更するモデル。Well-Architected 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

「オリジンアクセスコントロール」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

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

OI

「 オペレーションの統合」を参照してください。

OLA

「運用レベルの契約」を参照してください。

オンライン移行

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

OPC-UA

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

オープンプロセス通信 - 統合アーキテクチャ (OPC-UA)

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

オペレーショナルレベルアグリーメント (OLA)

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

運用準備状況レビュー (ORR)

インシデントや潜在的な障害の理解、評価、防止、または範囲の縮小に役立つ質問とそれに関連するベストプラクティスのチェックリスト。詳細については、 AWS Well-Architected フレームワークの「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するハードウェアおよびソフトウェアシステム。製造では、OT と情報技術 (IT) システムの統合が、Industry 4.0 変換の主な焦点です。

オペレーション統合 (OI)

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

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail をログに記録する、 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のため、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイド を参照してください。

オリジンアクセスコントロール (OAC)

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コントロールが可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」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

「プログラム可能なロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

アクセス許可を定義 (アイデンティティベースのポリシーを参照)、アクセス条件を指定 (リソースベースのポリシーを参照)、または の組織内のすべてのアカウントに対する最大アクセス許可を定義 AWS Organizations （ サービスコントロールポリシーを参照) できるオブジェクト。

多言語の永続性

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

ポートフォリオ評価

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

述語

`true` または を返すクエリ条件。一般的に`false`は `WHERE`句にあります。

述語プッシュダウン

転送前にクエリ内のデータをフィルタリングするデータベースクエリ最適化手法。これにより、リレーショナルデータベースから取得して処理する必要があるデータの量が減少し、クエリのパフォーマンスが向上します。

予防的コントロール

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

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM 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内にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通じてプライバシーを考慮するシステムエンジニアリングアプローチ。

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を防ぐように設計されたセキュリティコントロール。これらのコントロールは、プロビジョニング前にリソースをスキャンします。リソースがコントロールに準拠していない場合、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「 コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

設計、開発、発売から成長と成熟まで、製品のデータとプロセスのライフサイクル全体にわたる管理。

本番環境

「環境」を参照してください。

プログラム可能なロジックコントローラー (PLC)

製造では、マシンをモニタリングし、製造プロセスを自動化する、信頼性の高い適応可能なコンピュータです。

プロンプトの連鎖

1 つの LLM プロンプトの出力を次のプロンプトの入力として使用して、より良いレスポンスを生成します。この手法は、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善または拡張したりするために使用されます。これにより、モデルのレスポンスの精度と関連性が向上し、より詳細でパーソナライズされた結果が得られます。

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。

パブリッシュ/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。スケーラビリティと応答性を向上させます。たとえば、マイクロサービスベースの MES では、マイクロサービスは他のマイクロサー

ビスがサブスクライブできるチャネルにイベントメッセージを発行できます。システムは、公開サービスを変更せずに新しいマイクロサービスを追加できます。

Q

クエリプラン

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

クエリプランのリグレッション

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

R

RACI マトリックス

責任、説明責任、相談、通知 (RACI) を参照してください。

RAG

「取得拡張生成」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

責任、説明責任、相談、情報 (RACI) を参照してください。

RCAC

「行と列のアクセスコントロール」を参照してください。

リードレプリカ

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

再設計

「7 Rs」を参照してください。

目標復旧時点 (RPO)

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

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

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

回帰

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

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

プラットフォーム変更

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または回復するアプリケーションの機能。高可用性とディザスタリカバリは、 で回復性を計画する際の一般的な考慮事項です AWS クラウド。詳細については、AWS クラウド「レジリエンス」を参照してください。

リソースベースのポリシー

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

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

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

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、Implementing security controls on AWSのResponsive controlsを参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

取得拡張生成 (RAG)

LLM がレスポンスを生成する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を参照する生成 AI テクノロジー。たとえば、RAG モデルは、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する場合があります。詳細については、「RAG とは」を参照してください。

ローテーション

攻撃者が認証情報にアクセスすることをより困難にするために、シークレットを定期的に更新するプロセス。

行と列のアクセス制御 (RCAC)

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

RPO

「目標復旧時点」を参照してください。

RTO

目標復旧時間を参照してください。

ランブック

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

S

SAML 2.0

多くの ID プロバイダー (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 Management Console 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

「監視コントロールとデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存する AWS Secrets Managerパスワードやユーザー認証情報などの機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値は、バイナリ、単一の文字列、または複数の文字列にすることができます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの内容」を参照してください。

設計によるセキュリティ

開発プロセス全体でセキュリティを考慮するシステムエンジニアリングアプローチ。

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、予防的、検出的、応答的、プロアクティブの 4 つの主なタイプがあります。

セキュリティ強化

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

Security Information and Event Management (SIEM) システム

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

セキュリティレスポンスの自動化

セキュリティイベントに自動的に応答または修復するように設計された、事前定義されたプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動応答アクションの例としては、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の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、スループットなど、サービスのパフォーマンス側面の測定。

サービスレベルの目標 (SLO)

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

責任共有モデル

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

SIEM

セキュリティ情報とイベント管理システムを参照してください。

単一障害点 (SPOF)

システムを中断する可能性のあるアプリケーションの 1 つの重要なコンポーネントの障害。

SLA

「サービスレベルの契約」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

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、『』の「アプリケーションをモダナイズするための段階的アプローチ AWS クラウド」を参照してください。

SPOF

単一障害点を参照してください。

スタースキーマ

1 つの大きなファクトテーブルを使用してトランザクションデータまたは測定データを保存し、1 つ以上の小さなディメンションテーブルを使用してデータ属性を保存するデータベース組織構造。この構造は、データウェアハウスまたはビジネスインテリジェンスの目的で使用するように設計されています。

strangler fig パターン

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

サブネット

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

監視制御とデータ収集 (SCADA)

製造では、ハードウェアとソフトウェアを使用して物理アセットと本番稼働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潜在的な問題を検出したり、パフォーマンスをモニタリングしたりする方法でシステムをテストします。Amazon CloudWatch Synthetics を使用して、これらのテストを作成できます。

システムプロンプト

LLM にコンテキスト、指示、またはガイドラインを提供して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、ユーザーとのやり取りのルールを確立するのに役立ちます。

T

tags

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 枚のピザで養う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する行のグループに対して計算を実行する SQL 関数。ウィンドウ関数は、移動平均の計算や、現在の行の相対位置に基づく行の値へのアクセスなどのタスクの処理に役立ちます。

ワークロード

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

ワークストリーム

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

WORM

「Write Once」、「Read Many」を参照してください。

WQF

AWS 「ワークロード認定フレームワーク」を参照してください。

Write Once, Read Many (WORM)

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

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を利用する攻撃、通常はマルウェア。

ゼロデイ脆弱性

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

ゼロショットプロンプト

LLM にタスクを実行する手順を提供しますが、タスクのガイドに役立つ例 (ショット) はありません。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。