AWS ホワイトペーパー

AWS での DevOps の概要

AWS での DevOps の概要: AWS ホワイトペーパー

Table of Contents

AWS での DevOps の概要

公開日: 2020 年 10 月 16 日 (改訂履歴)

要約

今日、これまで以上に多くの企業が、お客様とのより深いつながりを築き、持続可能で永続的なビジネス価値を達成するために、デジタルトランスフォーメーションの取り組みを始めています。あらゆる形態と規模の組織が、かつてないスピードで変革に取り組むことで新しい市場に参入し、競合他社を混乱させています。このような組織にとって、イノベーションとソフトウェアの創造的破壊に焦点を当てて、ソフトウェアデリバリーを効率化することが重要になります。アイデアから生産までの時間を短縮し、スピードと俊敏性を優先している組織は、「ディスラプター」として今後の創造的破壊を牽引できる可能性があります。

次のデジタルディスラプターになるには考慮すべき要因がいくつかありますが、このホワイトペーパーでは、AWS プラットフォームでの DevOps と、組織がアプリケーションやサービスを迅速に提供するために利用できるサービスおよび機能に焦点を当てています。

はじめに

DevOps とは、高品質のアプリケーションやサービスを迅速に提供するための組織の能力向上に役立つ、文化、エンジニアリング上のプラクティスとパターン、およびツールの組み合わせです。DevOps を取り入れるうえでの手法として、継続的インテグレーション、継続的デリバリー、Infrastructure as Code、モニタリングとログ記録など、いくつかの重要なプラクティスがこれまでに出現しました。

このホワイトペーパーでは、DevOps ジャーニーの加速に役立つ AWS の機能と、DevOps への対応によって生じる、差別化につながらない面倒な作業を AWS のサービスで解消する方法について取り上げます。また、サーバーやビルドノードを管理せずに継続的インテグレーションと継続的デリバリーの機能を構築する方法と、Infrastructure as Code を活用して、一貫性のある再現可能な方法でクラウドリソースのプロビジョニングおよび管理を行う方法についても説明します。

- 継続的インテグレーション: ソフトウェア開発手法の 1 つであり、デベロッパーが自分のコード変更を定期的にセントラルリポジトリにマージすると、自動的に構築とテストが実行されます。
- 継続的デリバリー: ソフトウェア開発手法の 1 つであり、コード変更が発生すると、自動的に構築、テスト、および実稼働環境へのリリース準備が実行されます。
- Infrastructure as Code: バージョン管理や継続的インテグレーションなどのソフトウェア開発技術とコードを使用して、インフラストラクチャのプロビジョニングと管理を行う手法です。
- モニタリングとログ記録: これらにより、アプリケーションとインフラストラクチャのパフォーマンスが製品のエンドユーザーエクスペリエンスにどのように影響しているかを確認できます。
- コミュニケーションとコラボレーション: ワークフローを構築し、DevOps に関する責任を分散させることで、チーム間の距離を縮めるための手法です。
- セキュリティ: あらゆるチームに共通する関心事です。継続的インテグレーションと継続的デリバリー (CI/CD) のパイプラインと関連サービスを保護し、適切なアクセス制御許可を設定する必要があります。

これらの各原則を調べると、アマゾン ウェブ サービス (AWS) が提供するサービスと密接な関連があることがわかります。

継続的インテグレーション

継続的インテグレーション (CI) とは、デベロッパーがコードに対する変更を一元的なコードリポジトリに定期的にマージし、その後で構築とテストを自動的に実行するソフトウェア開発手法です。CI は、バグの迅速な発見と対処、ソフトウェア品質の向上、新しいソフトウェアアップデートの検証とリリースにかかる時間の短縮に役立ちます。

AWS では、継続的インテグレーション用に以下のサービスを提供しています。

トピック

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeArtifact

AWS CodeCommit

AWS CodeCommit は、プライベート Git リポジトリをホストする、安全性と拡張性に優れたマネージド型ソースコントロールサービスです。CodeCommit を使用すると、お客様が独自のソース管理システムを運用する必要がなくなり、ハードウェアのプロビジョニングやスケーリング、ソフトウェアのインストール、設定、オペレーションも必要ありません。CodeCommit では、コードからバイナリまであらゆるものを保存でき、Git の標準機能がサポートされているため、Git ベースの既存ツールとシームレスに連携できます。プロジェクトの参照、編集、コラボレーションを行うオンラインのコードツールとして CodeCommit を利用することもできます。AWS CodeCommit には次のようなメリットがあります。

コラボレーション - AWS CodeCommit は、共同でのソフトウェア開発向けに設計されています。コードのコミット、分岐、結合を容易に行えるため、常にチームのプロジェクトを簡単に管理できます。CodeCommit では、プルリクエストもサポートされています。これにより、コードの確認をリクエストして共同作業者とコードを検討できます。

暗号化 - AWS CodeCommit との間では、必要に応じて HTTPS または SSH を使ってファイルを転送できます。また、リポジトリは AWS Key Management Service (AWS KMS) によって、保管中もお客様専用のキーで自動的に暗号化されます。

アクセスコントロール - AWS CodeCommit では AWS Identity and Access Management (IAM) を使用して、誰がデータにアクセスできるかに加えて、いつ、どこで、どのようにデータにアクセスでき

るかを制御および監視します。CodeCommit は、AWS CloudTrail および Amazon CloudWatch を介したリポジトリの監視にも役立ちます。

高い可用性と耐久性 - AWS CodeCommit では、リポジトリが Amazon Simple Storage Service (Amazon S3) およびAmazon DynamoDB に保存されます。暗号化されたデータは、冗長性を確保し、複数の施設にわたって保存されます。このアーキテクチャによって、リポジトリデータの可用性と耐久性が高められています。

通知とカスタムスクリプト - リポジトリに影響を与えるイベントの通知を受け取ることができるようになりました。通知は Amazon Simple Notification Service (Amazon SNS) の通知として送信されます。各通知には、ステータスメッセージに加えて、その通知が生成される原因となったイベントが生じたリソースへのリンクも含まれます。さらに、AWS CodeCommit リポジトリのトリガーを使用すると、Amazon SNS によって通知の送信または HTTP ウェブフックの作成を行うことができます。選択したリポジトリイベントへの応答として AWS Lambda 関数を呼び出すこともできます。

AWS CodeBuild

AWS CodeBuild はフルマネージド型の継続的統合サービスです。このサービスを使用すると、ソースコードのコンパイル、テストの実行、デプロイ可能なソフトウェアパッケージの作成を行うことができます。ビルドサーバーのプロビジョニング、管理、スケーリングは不要です。CodeBuild では、GitHub、GitHub Enterprise、BitBucket、AWS CodeCommit、Amazon S3 のいずれかをソースプロバイダーとして使用できます。

CodeBuild ではスケーリングが継続的に行われ、複数のビルドを同時に処理できます。CodeBuild は、さまざまなバージョンの Microsoft Windows および Linux 用に、多様な事前設定済み環境を提供しています。お客様は、独自にカスタマイズしたビルド環境を Docker コンテナとして使用することもできます。CodeBuild は、Jenkins や Spinnaker などのオープンソースツールとも統合できます。

CodeBuild では、単体テスト、機能テスト、統合テストのレポートも作成できます。これらのレポートには、実行されたテストケースの数と合格または不合格の数が視覚的に表示されます。ビルドプロセスは Amazon Virtual Private Cloud (Amazon VPC) 内で実行することもできます。これは、統合サービスまたはデータベースが VPC 内にデプロイされている場合に役立つ可能性があります。

AWS CodeArtifact

AWS CodeArtifact は、フルマネージド型のアーティファクトリポジトリサービスです。組織はこのサービスにより、ソフトウェア開発プロセスで使用するソフトウェアパッケージを安全に保存、公開、および共有できます。CodeArtifact では、デベロッパーが最新バージョンにアクセスできるよう

に、ソフトウェアパッケージと依存関係をパブリックなアーティファクトリポジトリから自動的に取得するように設定できます。

昨今では、ますます多くのソフトウェア開発チームが、アプリケーションパッケージで一般的なタスクを実行するために、オープンソースパッケージを取り入れています。ソフトウェア開発チームにとっては現在、脆弱性がないオープンソースソフトウェアの特定のバージョンを継続的に管理することが重要になっています。CodeArtifact を使用すると、前述の動作を強制するようにコントロールを設定できます。

CodeArtifact は、一般的に使用されるパッケージマネージャーと
Maven、Gradle、npm、yarn、twine、pip などのビルドツールで動作するため、既存の開発ワークフローに簡単に統合できます。

継続的デリバリー

継続的デリバリーとは、ソフトウェア開発手法の 1 つで、コード変更が発生すると、自動的に本番環境へのリリース準備が実行されるというものです。モダンアプリケーション開発の柱となる継続的デリバリーは、継続的インテグレーションを拡張したもので、すべてのコード変更が、構築段階の後にテスト環境または運用環境 (あるいはその両方) にデプロイされます。適切に実装されていれば、標準化されたテストプロセスに合格してデプロイ準備の整ったビルドアーティファクトが常にデベロッパーの手元にある状態になります。

継続的デリバリーを使用すると、デベロッパーは単なる単体テストを超えたテストを自動的に実行できるため、お客様へのデプロイ前にさまざまな角度からアプリケーションの更新を検証できます。例えば、UI テスト、ロードテスト、統合テスト、API 信頼性テストなどを実行できます。これは、デベロッパーが更新をさらに徹底的に検証し、先を見越して問題点を発見するために役立ちます。オンプレミスでは従来困難だった、複数の環境をテスト用に作成してレプリケーションを行う作業も、クラウドを利用すれば、簡単かつコスト効率のよい方法で自動的に実行できます。

AWS では、継続的デリバリー用に以下のサービスを提供しています。

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

トピック

- AWS CodeDeploy
- AWS CodePipeline

AWS CodeDeploy

AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2)、AWS Fargate、AWS Lambda などのさまざまなコンピューティングサービスおよびオンプレミスサーバーへのソフトウェアのデプロイを自動化する、フルマネージド型のデプロイサービスです。AWS CodeDeploy を使用すると、新しい機能をすばやく簡単にリリースできます。また、アプリケーションのデプロイ時のダウンタイムを回避し、アプリケーションの複雑な更新を処理できます。さらに、CodeDeploy の使用によってソフトウェアのデプロイを自動化できるため、ミスが起こりやすい手動操作の必要がなくなります。このサービスは、デプロイのニーズに応じてスケールを調整できます。

CodeDeploy には、継続的デプロイの DevOps 原則に沿った次のようなメリットがあります。

自動デプロイ: CodeDeploy を使用すると、ソフトウェアのデプロイをすべて自動化し、デプロイの信頼性と速度を高めることができます。

コントロールの一元化: CodeDeploy では、AWS マネジメントコンソールまたは AWS CLI から簡単にアプリケーションのデプロイを開始し、ステータスを追跡できます。CodeDeploy では、各アプリケーションリビジョンがいつどこにデプロイされたかを表示できる詳細なレポートを参照できます。またプッシュ通知を作成して、デプロイについてのライブアップデートを受信できます。

ダウンタイムの最小化: CodeDeploy は、ソフトウェアデプロイプロセスの実行中にアプリケーションの可用性を最大化するために役立ちます。変更を段階的に導入し、設定可能なルールに従ってアプリケーションの正常性を追跡できます。エラーが発生した場合は、ソフトウェアのデプロイを簡単に停止してロールバックできます。

導入が容易: CodeDeploy はどのようなアプリケーションについても使用でき、プラットフォームや言語が異なってもエクスペリエンスは変わりません。このため、既存の設定コードを簡単に再利用できます。CodeDeploy は、既存のソフトウェアリリースプロセスや継続的デリバリーツールチェーン (AWS CodePipeline、GitHub、Jenkins など) と統合することもできます。

AWS CodeDeploy は複数のデプロイオプションをサポートしています。詳細については、「デプロイ戦略」を参照してください。

AWS CodePipeline

AWS CodePipeline は、ソフトウェアのリリースに必要なステップをモデル化、可視化、自動化できる継続的デリバリーサービスです。AWS CodePipeline では、コードの構築から、本番前の環境へのデプロイ、アプリケーションのテスト、本番環境へのリリースまで、リリースプロセス全体をモデル化できます。コードの変更が生じるたびに、AWS CodePipeline では、定義されたワークフローに従ってアプリケーションの構築、テスト、デプロイが行われます。リリースプロセスの任意のステージに APN パートナーツールまたは独自のカスタムツールを統合して、エンドツーエンドの継続的デリバリーソリューションを構築することもできます。

AWS CodePipelineには、継続的デプロイの DevOps 原則に沿った次のようなメリットがあります。

迅速なデリバリー: AWS CodePipeline を使用すると、ソフトウェアのリリースプロセスを自動化し、新機能をすばやくユーザーに提供できます。CodePipeline を使用すれば、フィードバックをすばやく反映し、より早くユーザーに新機能を届けることができます。

品質の向上: AWS CodePipeline では、構築、テスト、およびリリースのプロセスを自動化することで、一貫した品質チェックを通じてすべての新しい変更を適用できるため、ソフトウェア更新の速度と品質を向上させることができます。

容易に統合が可能: AWS CodePipeline は特定のニーズに合わせて簡単に拡張できます。リリースプロセスの任意のステップで、AWS の構築済みプラグインまたはお客様独自のカスタムプラグインを使用できます。例えば、GitHub からソースコードを取得する、オンプレミスの Jenkins ビルドサーバーを使用する、サードパーティーのサービスを使用してロードテストを実施する、カスタムオペレーションダッシュボードにデプロイ情報を送信するなどが可能です。

設定可能なワークフロー: AWS CodePipeline では、さまざまなステージのソフトウェアプロセスをコンソールインターフェイス、AWS CLI、AWS CloudFormation、または AWS SDK でモデリングすることができます。実施するテストは簡単に指定でき、アプリケーションおよび依存関係をデプロイするステップをカスタマイズすることもできます。

デプロイ戦略

デプロイ戦略では、ソフトウェアの提供方法を定義します。組織が使用するデプロイ戦略は、その組織のビジネスモデルによって異なります。完全にテスト済みのソフトウェアを提供する場合もあれば、開発中の機能 (ベータリリースなど) についてユーザーからの評価やフィードバックを求める場合もあります。次のセクションでは、さまざまなデプロイ戦略について説明します。

トピック

インプレースデプロイ

この戦略では、デプロイグループ内の各インスタンスでアプリケーションを停止し、アプリケーションの最新リビジョンをインストールして、新しいバージョンのアプリケーションを開始して検証することにより、デプロイを行います。ロードバランサーを使用すると、デプロイ中に各インスタンスの登録を解除し、デプロイ完了後にサービスに復元できます。インプレースデプロイは、サービスの停止を想定してすべてを一括で行うことも、ローリング更新として実行することもできます。AWS CodeDeploy と AWS Elastic Beanstalk では、インスタンスごと、半数ごと、1 回にすべてのデプロイ設定が用意されています。Blue/Green デプロイでも、インプレースデプロイと同じデプロイ戦略を使用できます。

Blue/Green デプロイ

Blue/Green (レッド/ブラックと呼ばれることもある) デプロイとは、同じアプリケーションの異なるバージョンを実行している同一の環境間でトラフィックを移行することによってアプリケーションを解放する手法です。Blue/Green デプロイでは、アプリケーションの更新中のダウンタイムを最小限に抑え、ダウンタイムと機能のロールバックに関するリスクを軽減できます。Blue/Green デプロイを使用すると、アプリケーションの新しいバージョン (Green) を古いバージョン (Blue) と一緒に起動し、トラフィックをそのバージョンに再ルーティングして問題の検出をロールバックする前に、新しいバージョンを監視およびテストできます。

Canary デプロイ

トラフィックを 2 回の増分で移行します。Canary デプロイは、Blue/Green 戦略に段階的アプローチを使用して、さらにリスク回避型にしたものです。このデプロイは、2 ステップ (リニア方式) で実行できます。まず新しいアプリケーションコードをデプロイして試用のために公開し、承認後に、環境内の残りの部分に (リニア方式で) ロールアウトします。

リニアデプロイ

リニアデプロイでは、毎回同じ間隔 (分) の等しい増分でトラフィックを移行します。増分ごとに移行するトラフィックの割合 (%) と、増分間の間隔 (分) を指定する、事前定義済みのリニアオプションから選択できます。

1 回にすべてデプロイ

1 回にすべてデプロイとは、元の環境からリプレース環境にすべてのトラフィックを一度に移行することを意味します。

デプロイ戦略のマトリックス

次のマトリックスは、Amazon Elastic Container Service (Amazon ECS)、AWS Lambda、および Amazon EC2/オンプレミスでサポートされているデプロイ戦略を示しています。

- Amazon ECS はフルマネージド型のオーケストレーションサービスです。
- AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。
- Amazon EC2 では、クラウド内で安全かつ自在に規模を変更できるコンピューティング性能を使用できます。

	A	B	C	D
1	デプロイ戦略のマトリックス	Amazon ECS	AWS Lambda	Amazon EC2/オンプレミス
2	インプレース	✓	✓	✓
3	ブルー/グリーン	✓	✓	✓*
4	Canary	✓	✓	X
5	線形	✓	✓	X
6	一括	✓	✓	X

> **Note**
>
> EC2/オンプレミスでのブルー/グリーンデプロイは EC2 インスタンスでのみ使用できます。

AWS Elastic Beanstalk のデプロイ戦略

AWS Elastic Beanstalk では次のタイプのデプロイ戦略がサポートされています。

- 一括: すべてのインスタンスでインプレースデプロイを実行します。

- ローリング: インスタンスをバッチに分割し、バッチのデプロイを 1 つずつ実行します。
- ローリングと追加バッチ: デプロイをバッチに分割しますが、最初のバッチでは既存の EC2 インスタンスにデプロイするのではなく、新しい EC2 インスタンスを作成します。
- イミュータブル: 既存のインスタンスを使用するのではなく、新しいインスタンスでデプロイする必要がある場合。
- トラフィックの分割: イミュータブルなデプロイを実行し、事前に決定された期間、一定割合のトラフィックを新しいインスタンスに転送します。インスタンスの正常な状態が維持されていれば、すべてのトラフィックを新しいインスタンスに転送し、古いインスタンスをシャットダウンします。

Infrastructure as Code

DevOps の基本原則として、デベロッパーがコードを扱うのと同じ方法でインフラストラクチャを扱う必要があります。アプリケーションコードには定義済みの形式と構文があります。プログラミング言語のルールに従ってコードを記述しないと、アプリケーションを作成できません。コードは、バージョン管理システムまたはソース管理システムに保存され、そこでコードの開発、変更、バグ修正の履歴が記録されます。コードのコンパイルや、アプリケーションへの組み込みを行う場合、一貫性のあるアプリケーションが作成され、ビルドには再現性と信頼性があることが期待されます。

Infrastructure as Code の実践とは、同じ厳密なアプリケーションコード開発をインフラストラクチャのプロビジョニングに適用することを意味します。設定はすべて宣言型アプローチで定義し、アプリケーションコードと同様に、AWS CodeCommit などのソース管理システムに保存する必要があります。また、インフラストラクチャのプロビジョニング、オーケストレーション、デプロイで、Infrastructure as Code の使用をサポートする必要があります。

従来、インフラストラクチャのプロビジョニングにはスクリプトと手動プロセスの組み合わせが使用されていました。これらのスクリプトは、バージョン管理システムに保存されている場合も、テキストファイルやランブックで手順ごとに記録されている場合もありました。多くの場合、ランブックの記述は、ランブックの使用またはスクリプトの実行とは別の人物によって行われます。スクリプトまたはランブックの更新頻度が十分でなければ、デプロイ時に致命的な問題が生じる可能性があります。そのような場合は、新しい環境の作成において、必ずしも再現性、信頼性、一貫性が保たれない結果になります。

これとは対照的に AWS では、DevOps に重点を置いたインフラストラクチャの作成と保守の方法を提供しています。ソフトウェアデベロッパーがアプリケーションコードを記述するように、AWS では、プログラムによって記述および宣言するという方法でインフラストラクチャの作成、デプロイ、保守を行うサービスを提供しています。このようなサービスを使用することで、厳密さ、明確さ、信頼性を確保できます。このホワイトペーパーで説明する AWS のサービスは、DevOps の手法の中核であり、AWS DevOps の高レベルの原則とプラクティスの基盤を形成しています。

AWS では、インフラストラクチャをコードとして定義するために以下のサービスを提供しています。

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- AWS Cloud Development Kit for Kubernetes

AWS CloudFormation

AWS CloudFormation は、デベロッパーが予測可能な方法で指定順序で AWS リソースを構築できるサービスです。リソースは、JSON (JavaScript Object Notation) 形式または YAML (Yet Another Markup Language) 形式を使用してテキストファイルに書き込まれます。テンプレートには、作成および管理するリソースのタイプに応じて、特定の構文と構造が必要になります。AWS Cloud9 など、任意のコードエディタを使用して JSON または YAML でリソースを記述し、バージョン管理システムにチェックインすると、指定したサービスが、安全かつ再現可能な方法で CloudFormation によって構築されます。

CloudFormation テンプレートはスタックとして AWS 環境にデプロイされます。スタックは、AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または AWS CloudFormation API を使用して管理できます。スタックで実行中のリソースについて変更を加える必要がある場合は、スタックを更新します。リソースに変更を加える前に、変更セットを生成できます。変更セットとは、変更案のサマリーです。変更セットを使用すると、実行中のリソース (特に重要なリソース) で変更によって生じる可能性のある影響を、変更の実施前に確認できます。

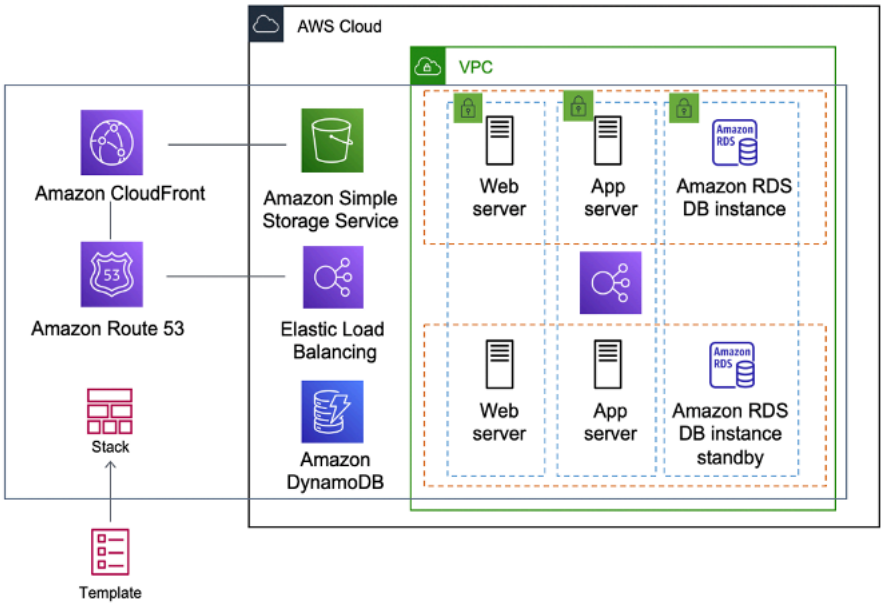

図 1 - AWS CloudFormation では 1 つのテンプレートワークフローから環境 (スタック) 全体を作成

1 つのテンプレートを使用して環境全体を作成および更新することも、別のテンプレートを使用して環境内で複数のレイヤーを管理することもできます。これによってテンプレートをモジュール化することが可能になり、多くの組織にとって重要なガバナンスのレイヤーも提供されます。

コンソールでスタックを作成または更新すると、設定のステータスを示すイベントが表示されます。エラーが発生した場合、デフォルトではスタックが前の状態にロールバックされます。イベン

トに関する通知は Amazon Simple Notification Service (Amazon SNS) によって処理されます。例えば、Amazon SNS を使用してスタックの作成と削除の進行状況を E メールで追跡し、プログラムによって他のプロセスと統合することもできます。

AWS CloudFormation を使用すると、AWS リソースのコレクションを簡単に整理およびデプロイできます。スタックを設定する際に、依存関係を記述することも、特殊なパラメータを渡すこともできます。

CloudFormation テンプレートを使用すると、Amazon S3、Auto Scaling、Amazon CloudFront、Amazon DynamoDB、Amazon EC2、Amazon ElastiCache、AWS Elastic Beanstalk、Elastic Load Balancing、IAM、AWS OpsWorks、Amazon VPC など、幅広い AWS サービスと連携できます。サポートされているリソースの最新リストについては、「AWS リソースおよびプロパティタイプのリファレンス」を参照してください。

AWS Cloud Development Kit

AWS Cloud Development Kit (AWS CDK) は、使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースのモデル化およびプロビジョニングを行うためのオープンソースソフトウェア開発フレームワークです。AWS CDK を使用すると、TypeScript、Python、Java、および .NET でアプリケーションインフラストラクチャをモデル化できます。デベロッパーは既存の統合開発環境 (IDE) を使用し、オートコンプリートやインラインドキュメントなどのツールを活用してインフラストラクチャの開発を加速できます。

AWS CDK では AWS CloudFormation をバックグラウンドで利用して、安全かつ再現可能な方法でリソースのプロビジョニングを行います。コンストラクトは CDK コードの基本的な構成要素です。1 つのコンストラクトは 1 つのクラウドコンポーネントを表しており、コンストラクトには、そのコンポーネントを構築するために AWS CloudFormation が必要とするすべてのものがカプセル化されています。AWS CDK には AWS Construct Library が含まれており、これには、多くの AWS サービスを表すコンストラクトが登録されています。コンストラクトを組み合わせることで、AWS にデプロイする複雑なアーキテクチャをすばやく簡単に作成できます。

AWS Cloud Development Kit for Kubernetes

AWS Cloud Development Kit for Kubernetes (cdk8s) は、汎用プログラミング言語を使用して Kubernetes アプリケーションを定義するためのオープンソースソフトウェア開発フレームワークです。

アプリケーションをプログラミング言語で定義すると (公開時点では Python と TypeScript のみがサポートされています)、アプリケーションの記述が cdk8s によって Kubernetes より前の YML に変換

されます。この YML ファイルは、任意の場所で実行されている Kubernetes クラスターで使用できます。この構造はプログラミング言語で定義するため、プログラミング言語から提供される豊富な機能を使用できます。プログラミング言語の抽象化機能を使用して、独自の定型コードを作成し、すべてのデプロイで再利用できます。

オートメーション

DevOps のもう 1 つの基本理念とプラクティスとなるものはオートメーションです。オートメーションでは、インフラストラクチャとそこで実行されるアプリケーションのセットアップ、設定、デプロイ、サポートが重視されます。オートメーションを使用すると、標準化された再現可能な方法で、より迅速に環境をセットアップできます。手動プロセスを排除することが、DevOps 戦略を成功させる鍵です。従来、サーバーの設定とアプリケーションのデプロイは、主に手動のプロセスを使用して行われてきました。このため環境が多様化し、問題が発生した場合は特定の環境を再現することが困難になります。

クラウドのメリットを最大限に引き出すには、オートメーションの使用が不可欠です。AWS の内部では、伸縮性およびスケーラビリティというコア機能を提供するために、オートメーションに大きく依存しています。手動のプロセスはエラーが発生しやすく信頼性に欠けるため、俊敏性を要するビジネスをサポートするには不適切です。組織では、手動設定を提供するために、高度なスキルを持つリソースが忙殺されることも少なくありませんが、その時間は本来、もっと重要で高い価値につながる活動のサポートに使う方が有効です。

最新の運用環境では一般的に、フルオートメーションを利用し、手動操作や本番環境へのアクセスを排除しています。これには、すべてのソフトウェアリリース、マシン設定、オペレーティングシステムへのパッチ適用、トラブルシューティング、バグ修正が含まれます。さまざまなレベルのオートメーションプラクティスを併用すると、より高度なエンドツーエンドの自動化プロセスを提供できます。

オートメーションには主に次のようなメリットがあります。

- 迅速な変更
- 生産性の向上
- 再現可能な設定
- 再現可能な環境
- 伸縮性の活用
- オートスケーリングの活用
- テストの自動化

オートメーションは AWS のサービスの基礎であり、すべてのサービス、機能、提供製品で内部的にサポートされています。

トピック

- AWS OpsWorks
- AWS Elastic Beanstalk

AWS OpsWorks

AWS OpsWorks では、AWS Elastic Beanstalk よりもさらに DevOps の原則が取り入れられています。これは、単なるアプリケーションコンテナではなく、アプリケーション管理サービスであると考えることができます。AWS OpsWorks では、設定管理ソフトウェア (Chef) との統合やアプリケーションライフサイクル管理などの追加機能により、さらに多様なレベルのオートメーションが可能になります。アプリケーションライフサイクル管理を使用すると、リソースのセットアップ、設定、デプロイ、アンデプロイ、またはシャットダウンのタイミングを定義できます。

AWS OpsWorks では、設定可能なスタックで独自にアプリケーションを定義でき、さらに高い柔軟性を実現しています。定義済みのアプリケーションスタックを選択することもできます。アプリケーションスタックには、アプリケーションサーバー、ウェブサーバー、データベース、ロードバランサーなど、アプリケーションに必要な AWS リソースのプロビジョニングがすべて含まれています。

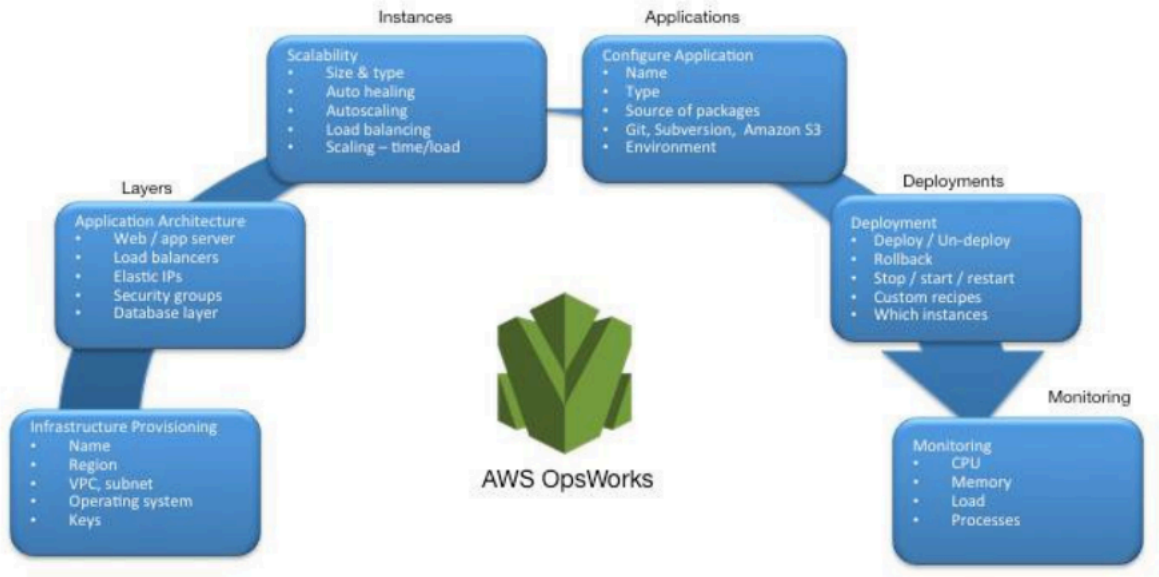

図 2 - DevOps の機能とアーキテクチャを示す AWS OpsWorks

アプリケーションスタックは、スタックを個別に管理できるように、アーキテクチャレイヤーno に編成されます。レイヤーの例としては、ウェブ層、アプリケーション層、データベース層などがあります。AWS OpsWorks を使用すると、追加設定なしで、Auto Scaling グループと Elastic Load

Balancing ロードバランサーのセットアップを簡素化できます。これは、DevOps 原則としてのオートメーションをよく表しています。AWS Elastic Beanstalk と同様に、AWS OpsWorks ではアプリケーションのバージョニング、継続的デプロイ、インフラストラクチャ設定管理がサポートされます。

AWS OpsWorks では、モニタリングとロギングの DevOps プラクティスもサポートされています (次のセクションで説明します)。モニタリングのサポートは Amazon CloudWatch によって提供されます。すべてのライフサイクルイベントがログに記録され、実行された Chef レシピと発生したあらゆる例外が、独立した Chef ログに記録されます。

AWS Elastic Beanstalk

AWS Elastic Beanstalk は、Java、.NET、PHP、Node.js、Python、Ruby、Go、Docker で開発されたウェブアプリケーションを、Apache、NGINX、Passenger、IIS などの一般的なサーバーに迅速にデプロイおよびスケーリングするサービスです。

Elastic Beanstalk は Amazon EC2 および Auto Scaling 上の抽象レイヤーであり、クローニング、Blue/Green デプロイ、Elastic Beanstalk Command Line Interface (eb cli) などの追加機能や、AWS Toolkit for Visual Studio、Visual Studio、Visual Studio Code、Eclipse、IntelliJ との統合を提供することで、デベロッパーの生産性を向上し、デプロイを簡素化します。

モニタリングとロギング

コミュニケーションとコラボレーションは DevOps の理念の基本です。これらを円滑に行うためには、フィードバックが非常に重要です。AWS では、Amazon CloudWatch と AWS CloudTrail の 2 つのコアサービスによってフィードバックが提供されます。これらの組み合わせによって、モニタリング、アラート、監査を行う堅牢なインフラストラクチャとなり、デベロッパーとオペレーションチームは緊密かつ透過的に連携できます。

AWS では、モニタリングおよびロギング用に以下のサービスを提供しています。

トピック

- Amazon CloudWatch
- Amazon CloudWatch アラーム
- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights
- Amazon CloudWatch Events
- Amazon EventBridge
- AWS CloudTrail

Amazon CloudWatch

Amazon CloudWatch メトリクスでは、Amazon EC2 インスタンス、Amazon EBS ボリューム、Amazon RDS DB インスタンスなど、AWS のサービスからデータが自動的に収集されます。これらのメトリクスは、ダッシュボードとして整理できます。さらに、アラームまたはイベントを作成してイベントをトリガーすることも、Auto Scaling アクションを実行することもできます。

Amazon CloudWatch アラーム

Amazon CloudWatch メトリクスによって収集されたメトリクスに基づいてアラームを設定できます。このアラームにより、Amazon Simple Notification Service (Amazon SNS) のトピックに通知を送信することも、Auto Scaling アクションを開始することもできます。アラームには、期間 (メトリクスを評価する時間)、評価期間 (最新のデータポイントの数)、アラームを実行するデータポイント (評価期間内のデータポイントの数) が必要です。

Amazon CloudWatch Logs

Amazon CloudWatch Logs は、ログを集約およびモニタリングするサービスです。AWS CodeBuild、CodeCommit、CodeDeploy、および CodePipeline を CloudWatch ログに統合して、すべてのログを一元的に監視することもできます。これらのサービスの他、さまざまな AWS サービスを CloudWatch と直接統合できます。

CloudWatch Logs では、次のことができます。

- ログデータの照会
- Amazon EC2 インスタンスからのログのモニタリング
- AWS CloudTrail でログに記録されたイベントのモニタリング
- ログ保持ポリシーの定義

Amazon CloudWatch Logs Insights

Amazon CloudWatch Logs Insights を使用すると、ログのスキャンにより、ログデータのインタラクティブな照会と可視化が可能になります。さまざまなログ形式を認識でき、JSON ログからのフィールドを自動検出できます。

Amazon CloudWatch Events

Amazon CloudWatch Events は、AWS リソースの変更を示すシステムイベントのストリームをほぼリアルタイムで提供します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。CloudWatch Events が発生すると、運用上の変更が認識されます。CloudWatch Events は、オペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、是正措置を実行します。

AWS のサービスの変更についてアラートを通知するルールを CloudWatch Events で設定し、Amazon EventBridge を使用してこれらのイベントを他のサードパーティシステムと統合できます。CloudWatch Events と統合されている AWS DevOps 関連のサービスを次に示します。

- Application Auto Scaling イベント
- CodeBuild イベント
- CodeCommit イベント

- AWS CodeDeploy イベント
- CodePipeline イベント

Amazon EventBridge

Amazon CloudWatch Events と EventBridge は同じように基盤となるサービスと API ですが、EventBridge の方が多くの機能を提供しています。

Amazon EventBridge は、AWS のサービス、Software as a Service (SaaS)、およびアプリケーションの間の統合を可能にするサーバーレスイベントバスです。イベント駆動型アプリケーションの構築に加え、EventBridge を使用すると CodeBuild、CodeDeploy、CodePipeline、CodeCommit などのサービスからのイベントを通知できます。

AWS CloudTrail

コラボレーション、コミュニケーション、透明性という DevOps の原則を取り入れるには、インフラストラクチャに誰が変更を加えているのかを理解することが重要です。AWS では、この透明性は AWS CloudTrail サービスによって提供されます。AWS のすべてのやり取りは、AWS CloudTrail によってモニタリングおよびロギングされる AWS API コールを通じて処理されます。生成されたログファイルはすべて、お客様が定義した Amazon S3 バケットに保存されます。ログファイルは、Amazon S3 のサーバー側の暗号化 (SSE) を使用して暗号化されます。すべての API コールは、ユーザーから直接送信されたものか、ユーザーに代わって AWS のサービスによって送信されたものかにかかわらず、ログに記録されます。CloudTrail ログは、オペレーションチームがサポート用に、セキュリティチームがガバナンス用に、財務チームが請求用に使用するなど、多くのグループが活用できます。

コミュニケーションとコラボレーション

DevOps 文化を組織に導入する場合も、DevOps 文化を変革するための話し合いを行う場合も、コラボレーションはアプローチの重要な部分を占めます。Amazon では、チームの考え方を変える必要があると気付き、2 枚のピザチームというコンセプトを採用しました。

トピック

2 枚のピザチーム

「私たちはチームの規模を、ピザ 2 枚で賄える人数以下に抑えることにしています」とベゾスは述べています。「これを 2 枚のピザチームのルールと呼んでいます。」

チームが小さければ小さいほど、コラボレーションの質は高くなります。また、ソフトウェアリリースの動きがかつてないほど速くなっていることからも、コラボレーションが非常に重要になっています。さらに、ソフトウェアを提供するチームの能力が、組織にとって競合他社との差別化要因になる可能性があります。新しい製品機能のリリースやバグの修正が必要な状況を想像してみてください。市場投入時間を短縮するには、これらの作業をできるだけ速く行う必要があります。また、ゆっくりとしたプロセスで変革を進めるのではなく、変更による効果を段階的に得るために、アジャイルなアプローチを使用する場合にも、コラボレーションが重要になります。

責任共有モデルに移行し、サイロ化された開発アプローチから脱却し始める場合には、チーム間のコミュニケーションも重要になります。これにより、チームに所有者の概念がもたらされ、チームの視点がエンドツーエンドとしての捉え方に変わります。チームから見た本番環境が、可視性のないブラックボックスであってはなりません。

共通の DevOps チームを構築する場合や、チームに 1 人以上の DevOps 用メンバーを置くアプローチの場合には、文化の変革も重要です。どちらのアプローチにおいても、チーム内における責任共有の必要性が生じます。

セキュリティ

DevOps トランスフォーメーションに取り組んでいる場合も、DevOps の原則を初めて取り入れる場合も、セキュリティは DevOps プロセスに統合されたものであると考える必要があります。これは、構築、テスト、デプロイの各段階に共通する関心事です。

AWS での DevOps のセキュリティについて説明する前に、AWS 責任共有モデルについて見ていきましょう。

トピック

AWS 責任共有モデル

セキュリティは、AWS とお客様の間の責任共有です。責任共有モデルの構成要素については、以下で説明します。

- AWS の責任「クラウドのセキュリティ」 – AWS は、AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャの保護について責任を負います。このインフラストラクチャには、AWS クラウドサービスを実行するハードウェア、ソフトウェア、ネットワーク、および施設が含まれます。
- お客様の責任「クラウド内のセキュリティ」 – お客様の責任は、お客様が選択した AWS クラウドサービスに応じて異なります。この選択によって、セキュリティに関する責任の一環としてお客様が行う設定作業の量が決まります。

この共有モデルでは、ホストオペレーティングシステムや仮想レイヤーから、サービスが運用されている施設の物理的なセキュリティまで、さまざまなコンポーネントのオペレーション、管理、制御が AWS で行われるため、お客様は運用上の負担を軽減できます。下の図は、お客様が構築環境のセキュリティについて理解するために重要です。

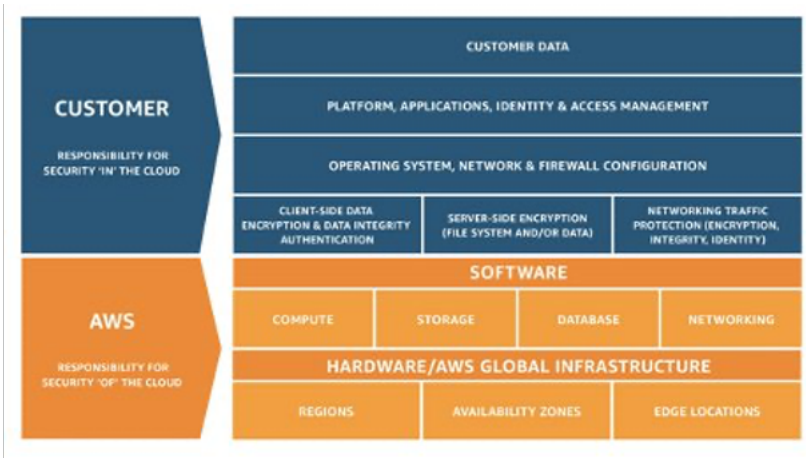

図 3 - AWS 責任共有モデル

Identity and Access Management

AWS Identity and Access Management (IAM) では、AWS リソースへのアクセスを管理するために使用するコントロールとポリシーを定義します。IAM を使用すると、ユーザーとグループを作成し、さまざまな DevOps サービスに対する許可を定義できます。

AWS リソースには、ユーザーだけでなくさまざまなサービスからのアクセスが必要になる場合もあります。例えば CodeBuild プロジェクトには、Docker イメージを Amazon Elastic Container Registry (Amazon ECR) に保存するためのアクセス権限と、Amazon ECR に書き込むためのアクセス許可が必要になる可能性があります。このようなタイプのアクセス許可は、サービスロールと呼ばれる特殊なタイプのロールによって定義されます。

IAM は AWS セキュリティインフラストラクチャのコンポーネントの 1 つです。IAM では、グループ、ユーザー、サービスロール、セキュリティ認証情報 (パスワードやアクセスキーなど) の他、ユーザーがアクセスできる AWS のサービスとリソースを制御するアクセス許可ポリシーを一元管理できます。IAM ポリシーでは、アクセス許可のセットを定義できます。このポリシーをロール、ユーザー、またはサービスにアタッチして、それぞれに対する許可を定義できます。また、IAM を使用して、適切な DevOps 戦略内で広く使用するロールを作成することもできます。場合によっては、許可を直接取得するのではなく、プログラムで AssumeRole を使用する方が適していることもあります。サービスまたはユーザーにロールを設定すると、通常はアクセスできないサービスにアクセスするための一時的な認証情報を割り当てることができます。

まとめ

テクノロジー企業がクラウドへの移行をスムーズ、効率的、かつ効果的に行うには、DevOps の原則とプラクティスを取り入れる必要があります。これらの原則は AWS プラットフォームに組み込まれています。事実、これらは多数の AWS サービス、特にデプロイおよびモニタリング関連サービスの基盤となっています。

まず、サービスとして AWS CloudFormation または AWS Cloud Development Kit (AWS CDK) を使用することにより、インフラストラクチャをコードで定義します。次に、AWS CodeBuild、AWS CodeDeploy、AWS CodePipeline、AWS CodeCommit などのサービスを利用し、アプリケーションで継続的デプロイを使用する方法を定義します。アプリケーションレベルでは、AWS Elastic Beanstalk、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS) などのコンテナと AWS OpsWorks を使用して、一般的なアーキテクチャの設定を簡素化します。これらのサービスを使用すると、Auto Scaling や Elastic Load Balancing など、他の重要なサービスの組み込みも容易になります。最後に、Amazon CloudWatch などの DevOps モニタリング戦略と、AWS IAM などの確かなセキュリティプラクティスを使用します。

AWS をパートナーにして DevOps の原則を取り入れると、ビジネスと IT 組織に俊敏性がもたらされ、クラウドへの移行を加速できます。

ドキュメントの改訂

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

update-history-change	update-history-description	update-history-date
欠落していた寄稿者セクションを復元しました	欠落していた寄稿者セクションとテキストの変更を復元しました	2020 年 11 月 21 日
新しいサービスが含まれるようにセクションを更新しました	新しいサービスが含まれるようにセクションを更新しました	2020 年 10 月 16 日
初版公開	ホワイトペーパーの初回公開日	2014 年 12 月 1 日

寄稿者

本書の寄稿者は次のとおりです。

- Muhammad Mansoor - ソリューションアーキテクト
- Ajit Zadgaonkar - ワールドワイドテックリーダー (モダナイゼーション)
- Juan Lamadrid - ソリューションアーキテクト
- Darren Ball - ソリューションアーキテクト
- Rajeswari Malladi - ソリューションアーキテクト
- Pallavi Nargund - ソリューションアーキテクト
- Bert Zahniser - ソリューションアーキテクト
- Abdullahi Olaoye – クラウドソリューションアーキテクト
- Mohamed Kiswani – ソフトウェア開発マネージャー
- Tara McCann – マネージャーソリューションアーキテクト

注意

お客様は、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。