GitLab

自動化されたソフトウェア配信によってDevSecOpsをより速くより簡単に実現する方法

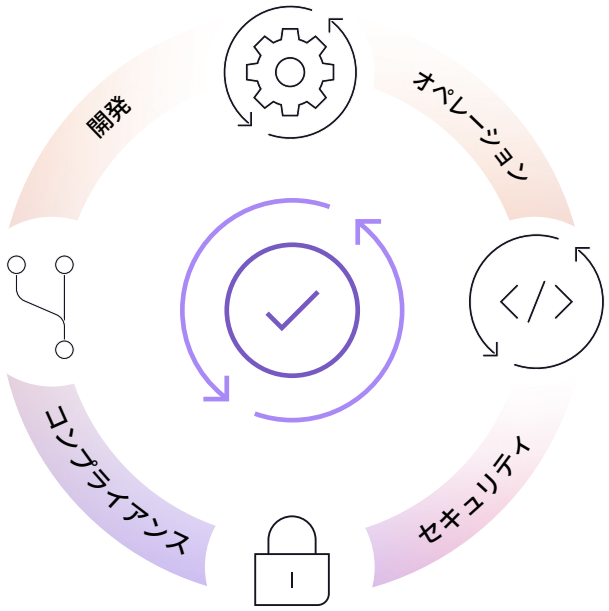
aws marketplace

はじめに

組織がクラウド移行とデジタルトランスフォーメーションの取り組みを競い合って進めるにつれ、開発チームと運用チームは大きなプレッシャーにさらされています。新たなビジネスチャンスを引き出すためにイノベーションを起こすことが求められているにもかかわらず、大規模かつモノリシックでもろく、互いに対話しないアプリケーション、コラボレーションしないサイロ化したチーム、あまりにも多くの手動プロセス、複数の複雑なツールが障壁となることがよくあります。GitLabが実施した2021年DevSecOpsグローバル調査によると、運用担当者のほぼ50%が、チームで2〜5種類のモニタリングツールを使用していると回答しました。[1]

解決策の1つは、最新のDevOps開発手法を採用し、開発と運用を継続的インテグレーション/継続的デプロイ(CI/CD)パイプラインにマージすることです。開発者の大多数は、DevOpsのおかげで、以前よりも2倍速くコードをリリースできていると述べており、パンデミック前である2021年から25%も増加しました。[2]しかし、クラウドネイティブのトランスフォーメーションとアプリケーションの近代化により成功を収めるには、従来のソースコード管理(SCM)とCIツールに依存する独立した手動プロセスから脱却し、価値実現までの時間を短縮する上で不可欠な自動化を実現する必要があります。

高品質のアプリケーションをより迅速かつ効率的に配信するためには、開発チーム、運用チーム、セキュリティチーム、コンプライアンスチームが、1つのインターフェイス、1つのプラットフォーム、および1つのデータモデルを通じて、オンデマンド環境を管理しながら、変更を継続的に管理、統合、検証、リリースできるツールを使用して連携し、最大限のスピードと効率を実現する必要があります。つまり、ソフトウェア開発ライフサイクルを自動化して、クラウドネイティブなKubernetesの導入を加速し、障害を減らして速度を上げ、反復的なタスクをなくすことで開発者の生産性を向上させる必要があります。



ヒント:高品質のアプリケーションをより迅速かつ効率的に配信するには、開発、運用、セキュリティおよびコンプライアンスの各チームが同じツールを使用して連携する必要があります。

CI/CDへの道のりにはスピードバンプが多数設置されています

目的地は分かりましたが、どのようにたどり着けばよいのでしょうか。DevOpsチームは、組織の成功に必要なシームレスなCI/CDパイプラインの実現を困難にするさまざまな課題に直面しています。これには以下が含まれます。

複雑さ:

開発者は複数の言語と複数のツールを習得し、複数の環境で作業しなければなりません。実際のところ、本来の仕事であるコーディングに集中し、ツールを切り替える必要がない状態を望んでいます。

クラウドへの移行:

アプリケーションをクラウドに移行するには、新しいパラダイムの中でアプリケーションをどのように動作させ、どのようにセキュリティを確保する必要があるかを再考する必要があります。

市場投入までの時間に関するプレッシャー:

モノリシックなアップデートや遅い手作業のプロセスにかける時間はありません。顧客の高い期待に応えるには、機能や修正を頻繁に提供する必要があります。

正確さとベストプラクティス:

時間に追われる開発者は、十分なテストを行わず、結果としてエラーや遅延が生じる可能性があります。

ガバナンス:

多くの場合、手作業のプロセスは文書化されておらず、監査証跡はなく、何か問題が発生したときにデプロイを元に戻すことは容易ではありません。

開発者の経験:

開発関連の人材はコストが高く、古い業界では特に確保が困難です。これは、小規模なチームほど生産性を高めなければならないことを意味します。一方で、開発者たちは暗黙の了解のようなタスクを負わされることを望んでいません。

セキュリティリスク:

IT環境が複雑になればなるほど、リスクを引き起こす可能性のある領域は大きくなり、開発者がデータやアプリケーションを安全に保護することは難しくなります。

設定:

下流の設定(ファイアウォール、ポリシーなど)は、しばしば別のシステムで保存・管理されるため、開発者が常にアクセスできるわけではありません。
開発者は、自分のアプリが影響を受けたり、完全にブロックされる可能性があることすら知らない可能性もあります。

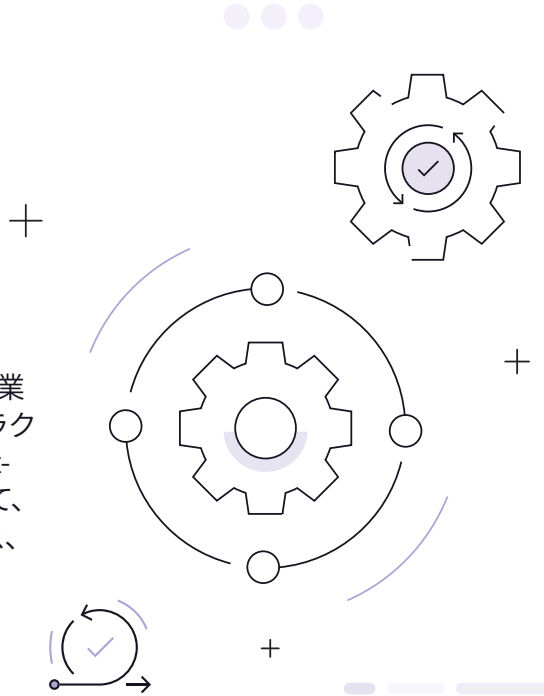
自動化されたソフトウェア配信によってこれらの課題を克服する方法

自動化されたソフトウェア配信(ソフトウェア開発パイプラインに含まれるプロセスをできるだけ多く自動化することを含む)の利点は複数あります。ClickOpsを排除し、クラウドネイティブの採用に不可欠なコントロールを導入することで、クラウドネイティブなKubernetesの採用をスピードアップできます。エラーを早期に検出し、リスクが軽減されるように、すべての変更でより多くのテストを行って配信可能にします。そして、最も重要な利点は、手動の作業からチームを解き放って生産性を向上させ、繰り返しの作業を最小限に抑えることで、価値を生み出す作業に集中できるようになることです。ソフトウェア配信の自動化を取り入れることで、DevOpsチームは次の3つの重要な方法で最も複雑な課題を克服できます。

01

継続的なインテグレーションと検証を可能にします。

コード、ビルド、テストを自動化し、変更が行われるごとに段階的に更新することで、チームは質の高いアプリケーションを大規模に構築できるようになり、デジタルトランスフォーメーションが加速されます。自動化により、エラーを早期に検出することでリスクが軽減され、並行ビルドとマージトレインを促進することで、ガバナンスを維持しながら開発者の生産性を向上させます。



02

複雑さを軽減します。

再現性のある安定した環境を自動的に作成できるため、クラウドでの作業の複雑さを大幅に軽減し、手作業によるインフラストラクチャ構成やClickOpsに伴うリスクを最小限に抑えることができます。インフラストラクチャを自動化すると、より迅速にリリースを行い、エラーからより迅速に回復することができます。また、git-opsによりアドオンとセキュリティポリシーの構成をモジュール化し、簡素化できます。さらに重要な点として、変更監査とガバナンスガードレールを自動化することで、より高いレベルのセキュリティ、コンプライアンス、説明責任を実現できることです。

03

継続的デリバリーを保証します。

アプリケーションのリリースプロセスを自動化することで、オンデマンドのソフトウェア配信を繰り返し実行できるようになります。すべての変更は「リリース可能」であるため、チームは変更を徐々に導入して混乱を最小限に抑え、変更を一部のユーザーでテストすることによってフィードバックを迅速に得ることができます。

自動化できるもの

最先端のDevOpsプラットフォームでは、ビルドから運用まで、ほぼすべてのプロセスを自動化できます。
以下にその一例を示します。

ビルド時:ビルド環境とリポジトリの基本的な構成を自動化することで、開発者は自分の専門分野、つまりアプリのコーディングに集中できるようになります。

- ✓ビルドの自動化には以下が含まれます。
- ✓サーバーとストレージの構成
- ✓インフラストラクチャコードとGitOps
- ✓ 複数環境のインフラストラクチャ構成ポリシー

テスト中:過去3年間、DevSecOps調査の回答者の過半数が、最も遅延が発生しやすい分野としてテストを挙げています。[3]テストの自動化によるメリットは非常に大きく、より速く、より正確かつ完全にテストを実行できるようになり、回転イス式管理を避けて、コンテキストスイッチが減少します。品質保証(QA)およびコンプライアンステストは、一度プログラムすれば、何度でも実行できます。また、AIはソースコードを自動検出してエラーをチェックすることも可能です。自動化されるテストジョブには、以下が含まれます。

- ✓静的分析と自動コードレビュー
- ✓コミットごとにコード品質とセキュリティを自動スキャン
- ✓コンテナ内のセキュリティ問題の特定
- ✓プロジェクトの依存関係とセキュリティ問題の分析
- ✓ライセンス依存関係のスキャン
- ✓認証情報と機密データ漏えいの検出
- ✓複数のプログラミング言語を対象としたセキュリティ分析の実行
- ✓言語とフレームワークを対象とした専用のユニットテスト
- ✓インテグレーションテスト
- ✓パフォーマンステスト

レビューと承認:テストが完了したら、仮環境をスピンアップするプロセスを簡単に自動化し、アプリケーションセキュリティテスト、コードの分析、潜在的なセキュリティ問題のチェックを実行できます。以下に例を示します。

- ✓自動コードレビュー
- ✓レビュアーの割り当てとルーティング
- ✓承認ルールおよび検証ルールと適用

本番環境:この最終段階では、アプリケーションのロールアウト中にユーザーエクスペリエンスが中断しないように、段階的なデプロイを自動化できます。

- ✓段階的なカナリアリリースまたはブルーグリーンデプロイ
- ✓自動ロールバック

GitLabとAWSを連携させてソフトウェア配信を自動化する方法

DevSecOpsコンピテンシーを持つ認定AWSアドバンスドテクノロジーパートナーとして、GitLabのCI/CDは、業界をリードするクラウドプラットフォームを使用して、顧客を成功に導いた実績のあるモデルです。

GitLabは、サイト信頼性エンジニアリング、AWS Elastic Kubernetes Service (EKS)クラスタープロビジョニングなどを含む、AWSでの一貫性と信頼性の高い継続的なデプロイを保証するためのプラットフォーム固有の実装パターンを多数提供しています。GitLabとAWSの緊密な統合は、すべてのワークロードのワークフローで、より優れたクラウドネイティブアプリケーションをより迅速に実現します。以下に例を示します。

仮想マシン:

Amazon Elastic Compute Cloud (EC2) は、スケーラブルなAWSクラウドコンピューティングキャパシティを提供し、GitLabが複数のマシン間でジョブをスケーリングできるようにします。Amazon Graviton 2インスタンスでGitLabを一緒に使用すると、インフラストラクチャのコストを大幅に削減できます。

Serverless:

GitLab上でワンクリックするだけで、AWS Fargateはスケーラブルなサーバーレスコンテナのデプロイを可能にします。AWS Fargateに移行することで、組織はインフラストラクチャ管理の労力を削減し、管理負担の軽減、コンピューティングリソースの最適化、インフラストラクチャ費用の削減を実現できます。

Kubernetes:

GitLab CI/CDでは、既存のAWSユーザーが他のAWSサービスや機能との緊密な統合を利用できる唯一のKubernetesサービスであるAWS EKSの統合クラスターを作成できます。また、GitLabは、ハイブリッド環境でAmazon EKS-Dをサポートしています。

イベント駆動型コンピューティング:

AWS Lambdaは、イベントに応答してコードを実行し、そのコードが必要とするコンピューティングリソースを自動的に管理するコンピューティングサービスです。GitLabは、AWS Lambda関数、サーバーレスアプリケーションスイッチであるAWS Serverless Application Model (AWS SAM)、およびGitLab CI/CDの開発をサポートします。

コンテナ:

GitLab CI/CDからAWSコマンドを実行し、AWS Elastic Container Services (ECS)上のGitLabのCIテンプレートでDockerのデプロイを自動化すると、時間を節約できます。

.Net:

GitLabを使用すれば、AWSで.Netアプリケーション用のCI/CDが可能になります。また、AWS LambdaまたはAWS Fargate上のGitLabを使用すると、サーバーレスリソースを含むコンテナ化されたアプリケーションを自動的にデプロイできます。

GitLab

AWSを利用するお客様は、2つの導入オプションのいずれかを選ぶことができます。GitLabは、AWS上のベアメタル、VM、コンテナなど、あらゆる環境で動作するGitLabインスタンスをインストール、管理、保守するGitLab Self-Managedと、インストール不要でサインアップしてすぐに使い始められるGitLab SaaSから選択できます。

GitLab on AWSを使用したソフトウェアの自動配信により、チームは手作業や反復作業を排除してソフトウェア開発ライフサイクルをスピードアップし、質の高いアプリケーションを大規模に配信して、開発、運用、セキュリティチーム間のコラボレーションを促進できます。GitLabではAuto DevOpsをご利用いただけます。これは、事前定義されたすぐに使用できるCI/CDテンプレートであり、所有しているソースコードを自動検出します。ベストプラクティスに基づいて、アプリケーションを自動的に検出、ビルド、テスト、デプロイ、およびモニタリングできるため、DevOpsチームはより速く、より効率的に、より費用対効果の高い方法でより多くのことを成し遂げることができます。

GitLab on AWSを使用したCI/CDの自動化の詳細については、AWS MarketplaceのGitLabにアクセスしてください。

[1, 2, 3] GitLab、2021年グローバルDevSecOps調査


GitLab