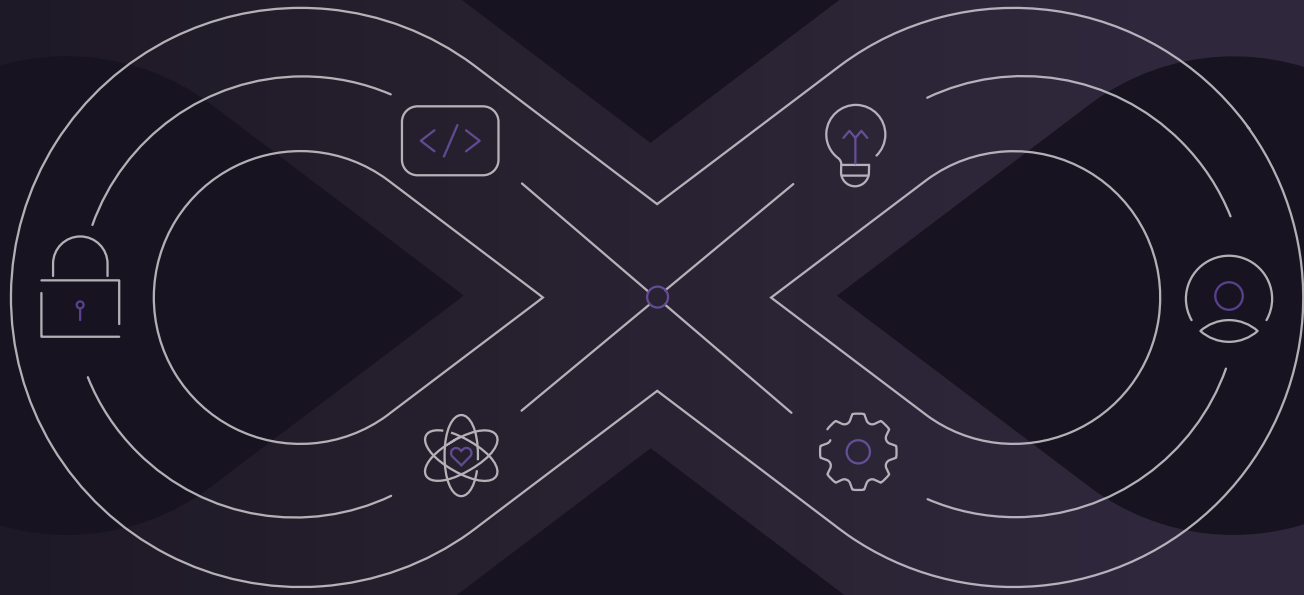
GitLab

DevOps ビギナーズガイド



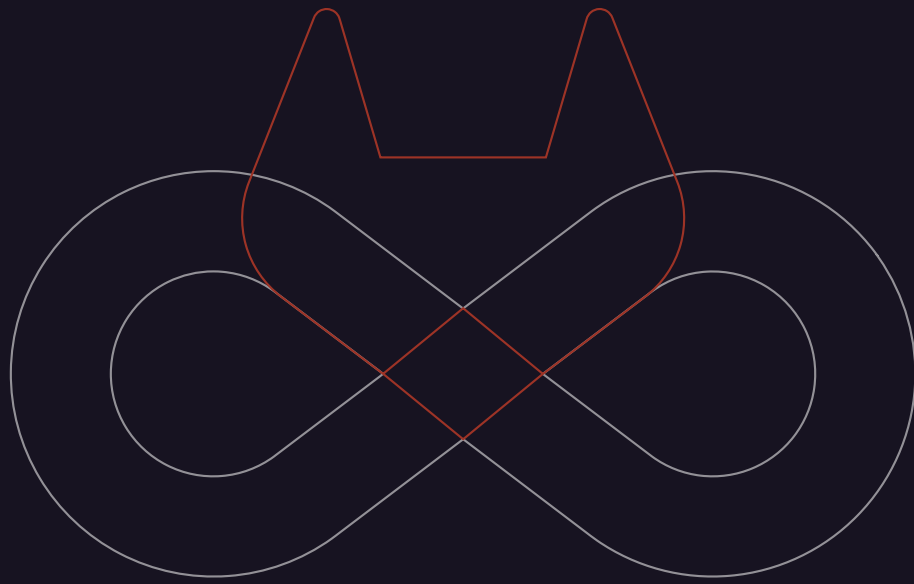
目次

はじめに

DevOps を使用している組織の数が劇的に増加している昨今、DevOps チームのメンバーが持つテクノロジーや実践方法についての知識が少ない、または全くないという事態には納得せざるを得ません。GitLab の **2023 年のグローバル DevSecOps 調査**によると、回答者の 56% が開発手法に DevOps または DevSecOps を使用していると報告しており、その割合は 2022 年の 47% から増加しています。

組織が最近 DevOps を採用した場合、または DevOps への切り替え準備をしている場合は、その裏にあるツールとプラクティスについての理解が完全でないことが考えられます。このガイドの目的は、これから DevOps について学ぼうとしている方々の学習をスピードアップできるようサポートすることです。DevOps の概要、鍵となるテクノロジー、理解すべき用語、そしてコラボレーションが重要な理由について解説していきます。また、DevOps の実用例を紹介しつつ、DevOps を学ぶことがキャリアアップにつながる理由も説明します。最後に、豊富なリソースの一覧を紹介します。

さあ、DevOps の世界に足を踏み入れましょう。



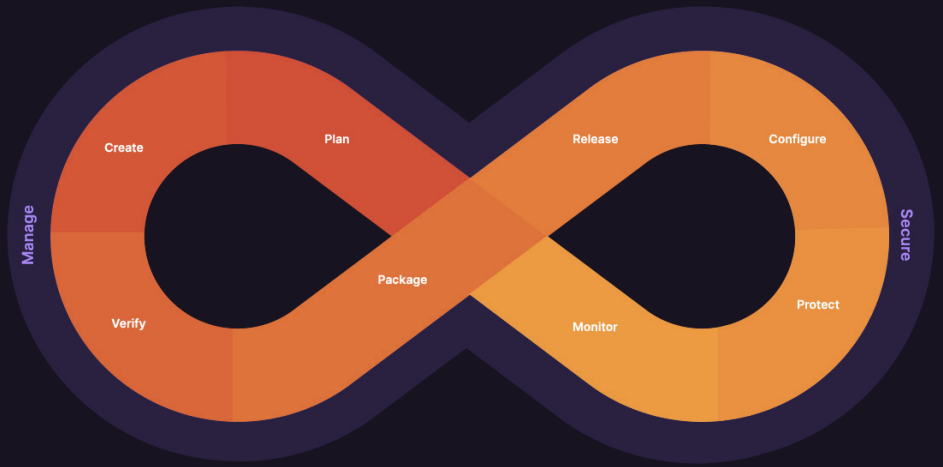
DevOps の概要と活用方法

DevOps について最初に知っておきたいのは、DevOps はチームを強化することに重点を置いているということです。組織が協働して安全なソフトウェアを開発し、より迅速かつ効率的に提供できるようサポートします。長年にわたり、ソフトウェア開発は合理的かつ効率的ではありませんでした。プロセスはサイロ化され、ボトルネックとコストのかかる遅れにつながり、セキュリティに取りかかるとしても、それは後回しとなっていました。従来のやり方に対する長年の不満から生まれた DevOps は、シンプルさとスピードを確実にもたらしています。

DevOps は単一のエンドツーエンドのプラットフォーム上で行われるのがベストだと、当社は考えています。チームがすべてが揃う単一 DevOps のプラットフォームソフトウェア開発エコシステムを使うことで、数が多く複雑で混乱しがちなツールからの移行や、使用が回避できたりします。こうすることで、チームメンバーが 1 つのツールから別のツールに移行する必要がなくなり、時間とコストの両方を節約できるのです。そのエコシステムを活用すれば、より優れた、よりコンプライアンスに準拠した、より安全なソフトウェアを、より効率的に、継続的に、最高速度で開発、構築、そして最終的に提供することができます。これにより、DevOps チームが**より俊敏**になるだけでなく、ビジネス全体にもアジリティがもたらされます。つまり、企業は顧客のニーズをより迅速に満たし、コンプライアンスに準拠し、競合他社をリードし、変化するビジネス環境を最大限に活用することができます。DevOps は、ビジネスだけでなく、ソフトウェア開発とデプロイメントの両方でアジリティを推進するエンジンなのです。

自動化の活用、セキュリティのシフトレフト、プロセスの再現性と測定可能性の向上により、DevOps プラットフォームはより優れたソフトウェアの開発を可能にし、質の高い新機能を設計して本番環境へ導入するまでにかかる時間を短縮します。これがソフトウェア開発全体のリターンを最大化するのです。

DevOps の仕組みについて説明するには、DevOps の背後にある**カルチャー**、つまり考え方について説明する必要があります。通常の開発ではありません。DevOps カルチャーの根幹は**コラボレーションと共同責任**です。そして、迅速なイテレーション、測定、評価、再評価の一定のサイクルに焦点を当てることです。繰り返しになりますが、アジリティ、そしてスピーディに学習してデプロイできることがすべてです。これらすべてが、継続的かつ反復的な改善と機能のデプロイにつながります。



基盤テクノロジーとプロセス

DevOps プロセスの各ステージ

計画から新機能のローンチ、分析、フィードバックの収集に至るまでのライフサイクルのステージについて理解しなければ、DevOps を学んだとは言えません。

DevOps に携わる以上、これらのフェーズはどれもプロセスの不可欠な部分であるため、しっかりと学んでおく必要があります。俯瞰的な目線で見ると、ビルド、テスト、デプロイという 3 つの包括的なステージが論理的な順序で実行されていることになります。これが自然なワークフローです。コードをビルドしてテストし、すべて上手くいけばデプロイします。

しかし、これらのステージは、さらに複雑に掘り下げる必要があります。一つひとつが、ソフトウェアとビジネスの価値を生み出す重要な原動力となるからです。このフローを理解して活用することで、効率性、信頼性、スピード、そしてアジリティを実現できます。では、9 つの重要なステージについてそれぞれ詳しく見ていきましょう。

- **計画**は、コードを書き始める前に行うすべてのことを包括する、DevOps のステージです。ここで重要なのは、今後の開発の指針となる製品ロードマップを作成して、チームがリソースと優先順位を整理して足並みを揃え、プロジェクトの進捗を把握できるようサポートすることです。
- **作成**は、CI/CD パイプラインの最初のステージです。ここでは、複数の開発者が同じコードベースに対して行う変更を調整するために、バージョン管理を使用したコードの設計と開発が行われます。これは速度を向上させるための重要な点の 1 つです。
- **検証**は、コードの品質を検証することに焦点を当てたプロセスです。開発者とテスターに迅速にフィードバックし、すべてのコミットに対するインサイトを即座に得るために、このプロセスは、セキュリティテスト、コード品質分析、並列実行、自動化に依存しています。開発者が開発中に欠陥を見つけて修正できるようにすることは、**より費用対効果が高く効率的である**ことが証明されています。
- **パッケージ**ステージは、コードが作成され、テストされた後に行われるステージです。アプリケーションと依存関係をパッケージ化し、コンテナを管理してアーティファクトを構築することで、一貫したソフトウェアサプライチェーンの維持が可能になります。
- **リリース**またはデプロイは、コードの更新を本番環境にプッシュすることです。DevOps では、イテレーションが作成され、テストされ、準備が整う度にリリースがデプロイされます。事前に計画された、固定の一括リリース日に行われるわけではありません。
- **構成**は、アプリケーション環境のセットアップ、管理、保守を行う段階です。自動化された構成管理は、サーバー、ネットワーク、ストレージシステムにまたがるこれらの複雑な環境を扱えるように設計されています。
- **監視**は、ソフトウェア、インフラストラクチャ、ネットワークの状態を追跡し、問題に対してアラートを発することに焦点を当てた、プロアクティブかつ自動化された段階です。これにより、セキュリティ、信頼性、アジリティが向上します。
- **保護**は、侵入や新たな脆弱性からアプリケーションとその実行環境を保護することを目的としています。
- **管理**は、権限の管理、DevOps のビルドやデプロイプロセスの標準化、ガードレールの自動化によるセキュリティおよびコンプライアンスポリシーへの確実な準拠により、エンドツーエンドのソフトウェア開発ライフサイクルを可視化し、管理することを目的とする段階です。

セキュリティの面はどうでしょうか？良い質問です。DevOps が優れているのは、セキュリティが後回しにされない点です。要件のドキュメント化から自動化されたテスト、その要件の検証まで、プロセスのすべてのステージにおいてセキュリティが考慮されています。これにより、新しいコードや機能が設計通りに動作すること、バグやセキュリティ上の脅威、コンプライアンス上の問題が生じないことを確実にできます。

これらのステージは、すべて継続的なサイクルの一環です。各ステージで作成されたすべての情報は、プラットフォームを通じて、ステージを越えてすべての参加者がすぐにアクセスでき、可視性とコラボレーションの向上につながる、唯一の情報源（Single Source of Truth）を提供します。また、ソフトウェア開発のライフサイクル全体を一元的に管理・コントロールできることも、統一されたプラットフォームの大きなメリットです。

これらの DevOps のステージを支えるもの

ソースコード管理（SCM）

コードのリポジトリは、ある人の変更によって他の人の変更が無効になることなく、多くの開発者間で共有されます。コードは、プロジェクトやプロジェクトのグループごとに分割されて管理されます。個々の開発者は、既存のコードをチェックアウトしたり、そこにあるものにコードを追加したりします。SCM ツールは、同じコードに対する編集の競合を特定し、解決のためにフラグを立てます。このプロセスにより、複数の開発者が 1 つのプロジェクトに同時に取り組むことができ、ソフトウェアアップデートのスピードを上げる鍵となります。DevOps は、その近代的なアーキテクチャが可能にする強力な機能により、**従来のバージョン管理システムとの重要な違いである** Git リポジトリを基盤としています。

継続的インテグレーション（CI）

このステップでは、共有のソースコードリポジトリに、開発段階の早期に頻繁に（1 日に何度も）変更をコミットし、それぞれの変更を自動的にテストしてビルドを開始することで、イテレーションが可能になります。

継続的インテグレーションは、効率性を重視しています。手作業を自動化し、より高い頻度でコードをテストすることにより、チームは迅速にイテレーションを行い、より少ない頻度で新機能をデプロイすることができます。**CI の他のメリット**としては、問題の特定と修正が容易になること、チームにとってコンテキストの切り替えが少ないこと、ユーザーと顧客がより良い結果を得られることなどがあります。

継続的インテグレーションを最大限活用するために、セットアップに以下の重要な要素が含まれていることを確認してください。

- ビルドを作成するのに必要なすべてのファイルとスクリプトを含んだソースコードリポジトリ。
- 単一コマンドからのビルドに必要なすべてを含んだスクリプトを備えた、自動化されたビルド。
- ポリシー（いずれかのテストが失敗した場合は失敗、など）を自動化するセルフテストビルド。
- 潜在的な競合を減らすための頻繁なコミットとイテレーション。
- 本番環境を正確に反映した、安定したテスト環境。
- 全開発者が最新の実行ファイルにアクセスでき、リポジトリに加えられたすべての変更を確認できる可視性。

継続的デリバリー（CD）

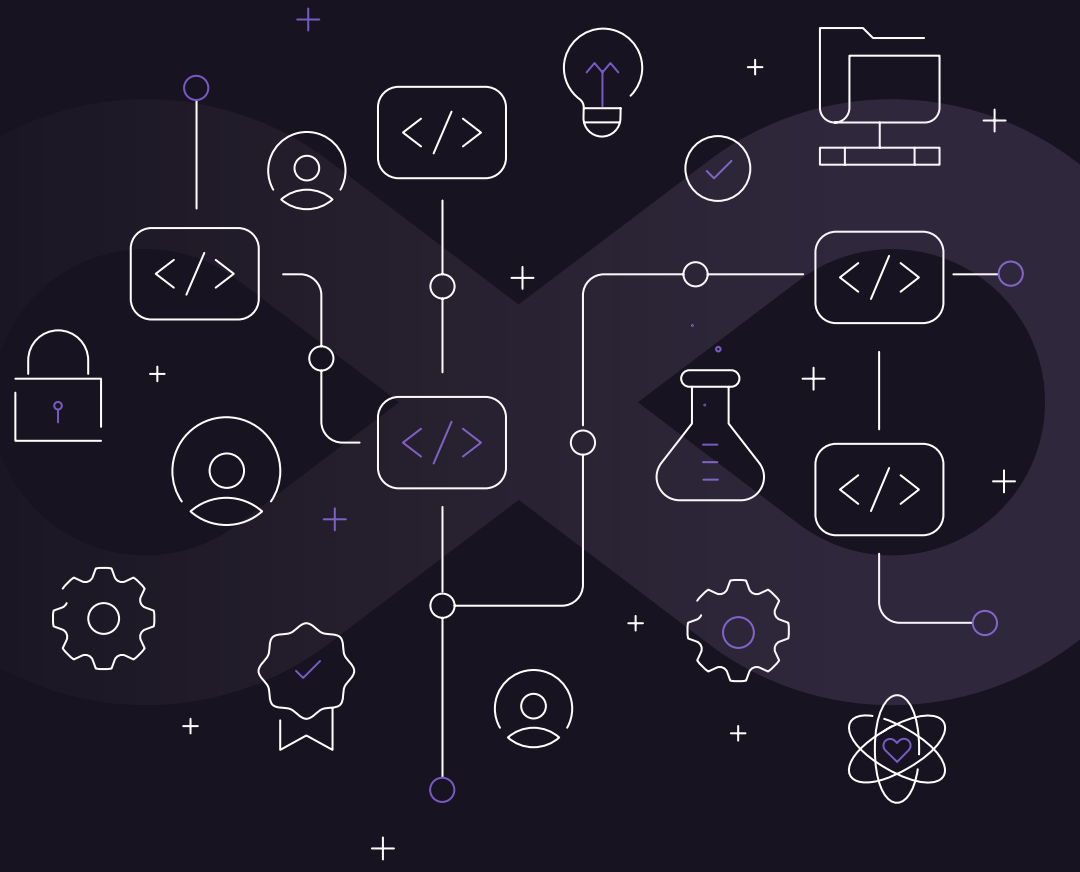
継続的デリバリーとは、継続的インテグレーションと連動してアプリケーションのリリースプロセスを自動化するソフトウェア開発プロセスのことです。CI プロセスの一環としてコードのテストとビルドが行われると、最終ステージで継続的デリバリーがそれを引き継ぎ、いつでも、どのような環境にもデプロイできるよう、必要なものすべてを含めてコードをパッケージ化します。継続的デリバリーは、インフラ環境のプロビジョニングから、テスト済みアプリケーションのテスト / ステージング環境または本番環境へのデプロイまで、すべてに対応できます。

継続的デリバリーでは、いつでも本番環境にデプロイできるようにソフトウェアが構築されます。手動でデプロイメントを開始するか、プロセスを自動化することができます。

継続的デリバリーが上手くいくと、ソフトウェアのリリース**プロセスは退屈なものになります。**つまり、リスクが少なく、一貫性があり、再現性があるプロセスが実現するということです。その後は自信を持ってリリースプロセスとスケジュールを計画し、インフラとデプロイメントを自動化し、クラウドのリソースをより効果的に管理することが可能になります。

自動テスト

これは、DevOps と継続的インテグレーションをフルに導入し、より高品質なコードを高い頻度でリリースするための鍵となります。CI パイプラインに組み込まれたテストでは、コミットされたあらゆるコード変更がビルドのトリガーとなります。トリガーされるとビルドはテストを実行し、加えられた変更が、アプリケーションに対して確立したすべてのテスト、ポリシー、およびコードコンプライアンス標準に合格するようにします。これにより、バグを早期に発見して容易に解決することができ、チームは高い頻度で、自信を持ってデプロイできるようになります。結果として、手動テストやプロセス後半での手直しを最小限に抑えられます。



セキュリティのシフトレフト

DevOps を成功させるための基本的なプロセスは、エンドツーエンドの自動化にセキュリティを組み込むことです。これは、多くの場合 DevSecOps と呼ばれます。ソフトウェア開発ライフサイクルの早い段階でテストとセキュリティレビュープロセスを統合することで、セキュリティ問題に適切に対処するための機会がより多くなります。セキュリティテストが後回しにされたり、コードが本番環境で実行されるまでテストが行われない場合、過去に遡って問題を修正することが難しくなり、迅速かつ効率的な問題修正ができなくなることがよくあります。その結果、デプロイの遅延、本番環境における脆弱性、技術的負債の増加、さらにセキュリティチームと他の DevOps チームとの間で非効率的なサイロが発生する可能性があります。

セキュリティをシフトレフトするには、CI パイプラインにセキュリティテストを統合し、共有リポジトリの他のコミットに対してだけでなく、全体的なセキュリティのためにコードを継続的にテストするのが良いでしょう。開発ライフサイクルの早い段階で実施するべきセキュリティテストには、次のようなものがあります。

- 静的アプリケーションセキュリティテスト（SAST）
- 動的アプリケーションセキュリティテスト（DAST）
- コンテナスキャン・クラスタ画像スキャン
- 依存関係スキャン
- シークレット検出
- Infrastructure-as-code（IAC）スキャン
- API テスト

ドキュメント

見落とされがちですが、DevOps の実践を成功させるには、ドキュメントが非常に重要です。チームが取り組むサービスやアプリケーション、そして DevOps プロセスの内容についての社内ドキュメントを作成し、管理することは、チームのパフォーマンスと提供するソフトウェアの質を向上させる上で大きな役割を果たします。2021 Accelerate State ofDevOpsReport によると、質の高いドキュメントを作成するチームは、ソフトウェアデリバリーと運用パフォーマンスが向上する可能性が 2.4 倍高いことが示されています。

フィードバック

組織は常にユーザーエクスペリエンスと DevOps プロセス全体を改善する方法を模索する必要があるため、フィードバックはソフトウェア開発にとって無くてはならないものです。従来のソフトウェア開発では、フィードバックループが複雑になることがありました。DevOps では、より緊密なコラボレーションと迅速なイテレーションにより、チームは管理可能なデータに常にアクセスできます。これにより、フィードバックを取り入れ、取り組みを調整し、効率的かつ迅速に改善を行うことが可能になります。プロセスの自動化は、情報を収集してチーム内の適切な部署に分散し、コードの更新に迅速に対応するための重要なステップです。

理解しておきたいその他の DevOps の重要テクノロジー

DevOps をして作業速度を向上するには、自己研鑽し、常に最新情報を得るべき分野がいくつかあります。以下の分野について理解を深め、最新情報をチェックしましょう。

クラウド

DevOps に携わっているのであれば、**クラウド**について理解することをお勧めします。現代のソフトウェアの大部分は、クラウドや、コンテナ、オーケストレータなどのクラウドネイティブなインフラに依存しており、これらのインフラはソフトウェア開発・デリバリープロセスの自動化に役立ちます。この方法で開発されたアプリケーションなら、DevOps の専門家はどこにでもデプロイでき、**複数のクラウドプラットフォーム**、あるいは自社データセンター内のプライベートクラウドを最大限活用できます。

クラウドネイティブなアプローチは、使用するハードウェアからコードを分離するため、よりスケーラブルです。DevOps の専門家は、会社に貢献し、自身のキャリアを築くために、クラウドプロバイダ、サービス、プラットフォームを理解する必要があります。そして、この分野は今、多くの DevOps 専門家に注目されています。**GitLab の 2023 年グローバル DevSecOps 調査**によると、クラウドコンピューティングは 2023 年の組織の投資の最優先事項であり、71% の回答者がアプリケーションの 4 分の 1 以上をクラウド上で実行していると答えています。

コラボレーション文化

コラボレーションは、DevOps の手法と哲学の根底にあるものです。経験豊富なメンバーと新しいメンバーの両方から議論、インプット、支援を求め、他者の専門知識から学び、それを頼りにすることでコラボレーション文化を構築できます。また、DevOps 文化を本当の意味で実践するには、他者の言葉に耳を傾け、プレッシャーの中で冷静に行動し、チームメンバー間の信頼を築き、状況や問題に対する主体性を持つことが求められます。

しかし、DevOps のチームメンバーが協働することだけがコラボレーションではありません。DevOps と他のビジネス部門（セキュリティ、マーケティング、財務、カスタマーサービス、経営陣）のコラボレーションも重要です。たとえば、DevOps とセキュリティチームとのコラボレーションにより、開発プロセス全体にセキュリティを組み込むことができます。これを、技術的なスキルよりも重要ではないソフトスキルとして片付けるという間違いを犯さないでください。コミュニケーションのような**重要なスキル**を伸ばし、ビジネスのニーズを伝える方法を知り、問題を解決するために協力するために必要です。2023 年のグローバル DevSecOps 調査では、39% の回答者が、業界のプロフェッショナルにとってコミュニケーションとコラボレーションのスキルが同じくらい重要であると答え、対象分野の専門知識（32%）とプログラミング（32%）を上回っています。

主要なプログラミング言語

DevOps エンジニアにはコーディングスキルが求められますが、さらに重要なのは、上記のエンドツーエンドの DevOps ライフサイクルのステージで使用されるプロセス、ツール、手法を考慮することです。プログラミング言語の中には、エンドツーエンドのプロセスに適したものとそうでないものがあります。数多くの言語が存在するため、何から手をつければいいかわからなくなるかもしれません。まず、DevOps チームが何を必要としているのかを理解する必要があります。どのようなプロジェクトがあり、現在どのような言語が必要とされるのでしょうか？また、今後のプロジェクトではどうでしょうか？

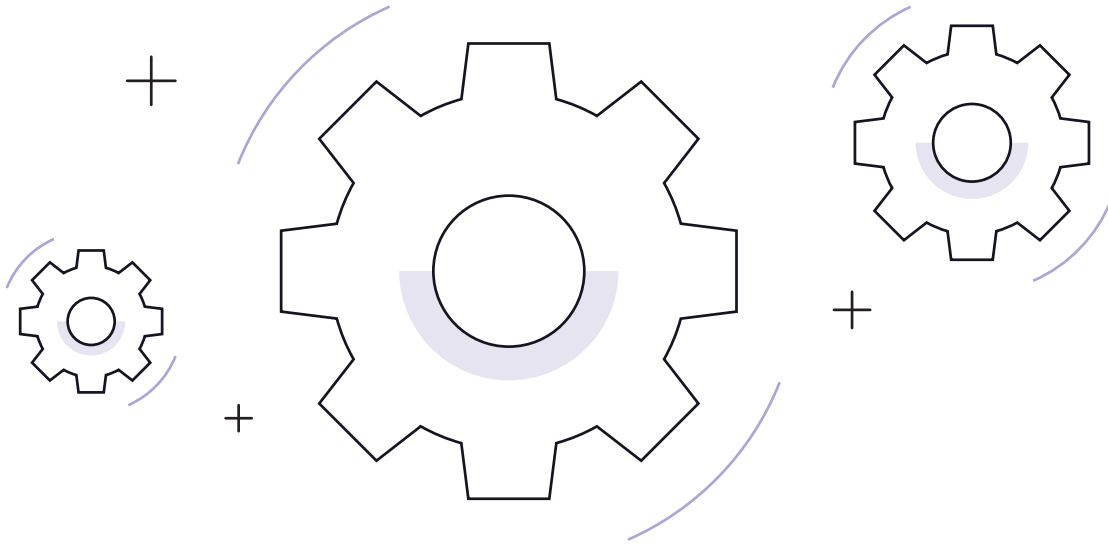
最も人気のあるプログラミング言語には、Python、Golang、Ruby、JavaScript、Perl、Java、Bash、PHP などがあります。**Stack Overflow 2022 開発者調査**によると、JavaScript は 10 年連続で最も一般的に使用されているプログラミング言語であり、また、HTML/CSS、JavaScript、Python は、コーディングを学ぶ人々の間で最も人気のある言語としてほぼ同率でした。

練習が必要なら、**ボランティアでコーディング**をしてみませんか。もしくはオープンソースプロジェクトに貢献して履歴書の内容を充実させることもできます。

自動化

DevOps チームは、開発およびデプロイのライフサイクル全体を通してプロセスを自動化することにますます重きを置いています。そのため、自動化の仕組みとその活用方法を理解しておく必要があります。自動化により、繰り返しの作業に使われる時間とコストが削減され、ヒューマンエラーがなくなることで、DevOps プロセス全体の効率が向上します。自動化を導入することで、各タスクが同じように、一貫性、信頼性、正確性を持って実行され、プロセスを加速し、納期を早め、最終的にはデプロイメントの数を増加させることができます。自動化によって人間の作業がなくなるわけではありません。しかし、可用性、パフォーマンス、セキュリティ問題のモニタリング、ソフトウェア環境の一貫した設定、事前に定義された品質基準に照らした新規アプリケーションバージョンのテスト、コードの統合、デプロイの高速化、開発プロセス全体を通した CI/CD テストソフトウェアの支援、ログとドキュメントの管理など、繰り返しの作業を管理する上での人間への依存度を最小限に抑えることができます。本当にたくさんのことができるのです。

2023 Global DevSecOps Survey によると、回答者の 63% が、ソフトウェア開発ライフサイクルを「完全に」または「ほとんど」自動化していると回答しています。また、32% が「やや」自動化していると答えています。この数字から、DevOps に携わるのであれば自動化について学ぶ必要があるということがわかります。



モニタリング

組織のアプリケーションスタックとそれに携わる DevOps チームの数が増えるにつれて、プロジェクトの数も増加の一途を辿っています。これらをすべて把握するのは大変なことです。そのため、エコシステムの状況をエンドツーエンドで、かつリアルタイムに把握するには、継続的なモニタリングが必要です。DevOps なら、複雑なアプリケーションを毎日、あるいは一日に何度も更新してデプロイすることができます。高度な自動モニタリングにより、バグの減少、デプロイの速度と効率の向上、セキュリティ脅威やコンプライアンス問題の検出、破壊的変更の排除、ドキュメントの管理などを積極的に行うことが可能になります。自動化は、計画から開発、統合、テスト、デプロイメント、そして運用に至るまで活用できます。

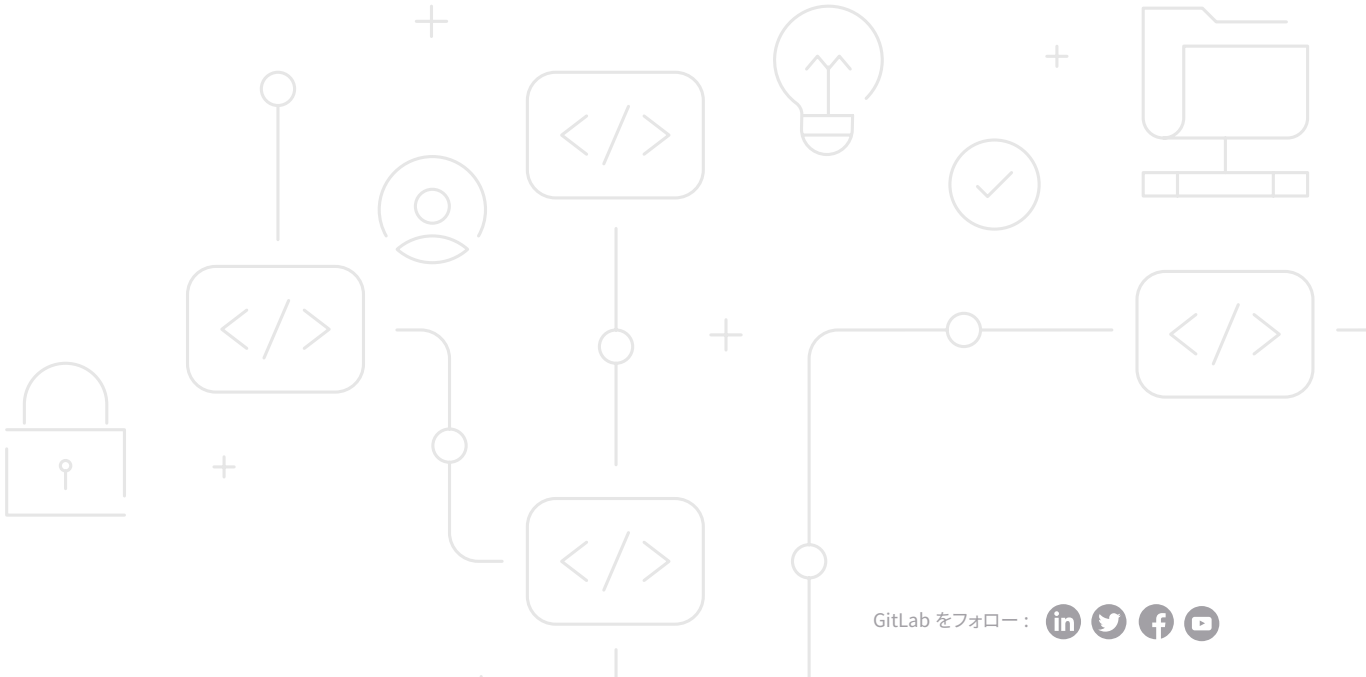
つまり、モニタリングは開発者だけでなく、プロジェクトリーダーやセキュリティチームにも必要なプロセスなのです。モニタリングは、プロセスの追跡だけでなく、パイプライン全体のパフォーマンスと脅威についてアラームを発するように設定されています。

高い目標を目指す DevOps チームは今後ますますモニタリングを活用する機会が増えるため、これについて理解しておく必要があります。

コンテナ

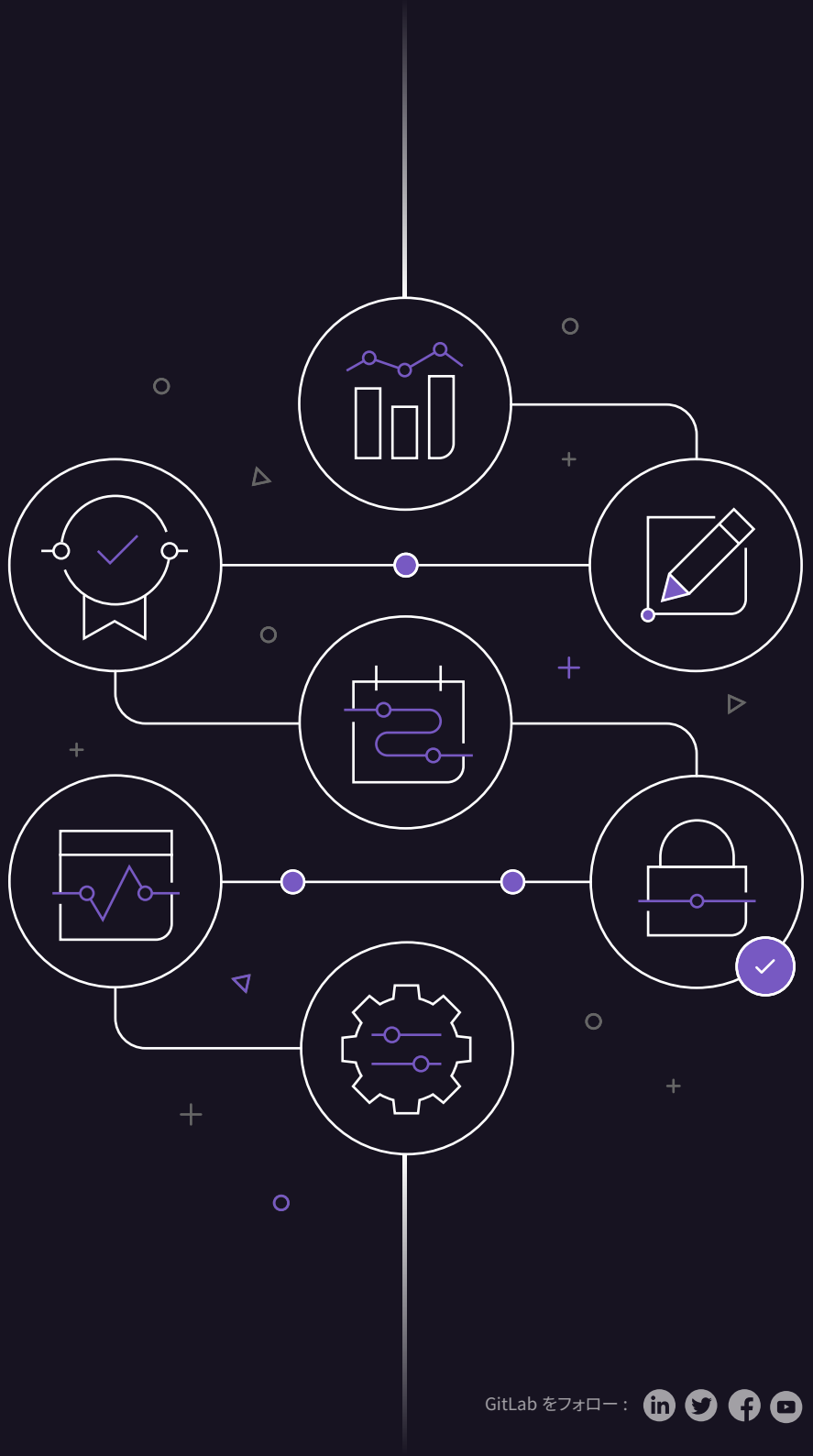
DevOps の世界では、基本的にソフトウェアのコード、その構成、システムライブラリ、ランタイム、およびその他の環境をパッケージ化したコンテナが広く採用されています。コンテナは、アプリケーションの実行に必要なすべてのものを備え、ある環境で構築されたアプリケーションが他の環境でも一貫してシームレスに動作することを保証し、環境間を移動する際にソフトウェアを確実に動作させます。これらのモジュールユニット（ビルディングブロック）は、DevOps チームが、限られたリソースでも最速で、効率的かつ安全にアプリケーションを構築、テスト、デプロイ、メンテナンスできるように設定されています。

これらはチーム間で簡単に共有できるため、開発やデプロイを迅速に行えるだけでなく、DevOps の鍵となるコラボレーション文化を推進することにもつながります。また、コンテナに関する知識を深めるには、一般的なアプリケーションコンテナ技術である Docker や、コンテナの実行方法・場所を制御するオープンソースのコンテナオーケストレーションシステムである Kubernetes について理解する必要があります。



DevOps による現実世界の問題の解決

DevOps の導入が企業にとってどれほど重要で変革的であるかを正確に把握するには、DevOps プラットフォームが HackerOne にもたらした効果を確認してください。世界で最も信頼されている、ハッカーを活用したセキュリティプラットフォームである HackerOne は、地球上最大のハッカーコミュニティへのアクセスを組織に提供します。グローバルに 70 以上の拠点を展開する同社は、部門間のコラボレーションが困難であることに気付きました。たとえば、異なる大陸で働く開発者が、他の開発者がコードプロジェクトで中断した部分を引き継がなければならなかった場合、長いパイプライン時間によって引き継ぎが中断されました。エンジニアリングチームは 3 倍に増えたため、HackerOne は開発とデプロイを迅速に行い、ツールチェーンの複雑さを軽減し、チームが複数のプロジェクトを効率的に管理できるようにする必要がありました。**彼らは DevOps プラットフォームで解決策を見つけることができたのです**。単一の統合された DevOps プラットフォームにより、HackerOne のチームはパイプラインの早い段階でコードの問題を発見し、反復的にセキュリティ上の欠陥を解決し、監査を簡素化することができました。また、デプロイを 1 日 1 ～ 2 回から 1 日最大 5 回に増やし、各開発者の開発時間を毎週 4 時間短縮しました。



リソース

ここでは、DevOps に関するリソースの一部をご紹介します。

DevOps の知識を深めるためのポッドキャスト

- **The Humans ofDevOpsPodcast Series**：スキルアップ、DevOps のしくみ、DevOps 分野で働く女性たちなどについてのインサイトが得られます。
- **ArrestedDevOps**：DevOps の最新動向について、トップ技術者たちの話が聞けます。
- **Real WorldDevOps**：DevOps カンファレンスの主催者、関連書籍の著者、エンジニア育成者との対談が聞けます。
- **The Cloudcast**：クラウド関連のあらゆるトピックに焦点を当てています。
- **Greater Than Code**：DevOps、およびテック分野全般における人的・技術的問題の双方を取り上げています。
- **Code Newbie Podcast**：ソフトウェア開発を始めたばかりの人を対象にしています。
- **DevOpsParadox**：業界の著名人による DevOps の解説が聞けます。

役立つ書籍・eBook

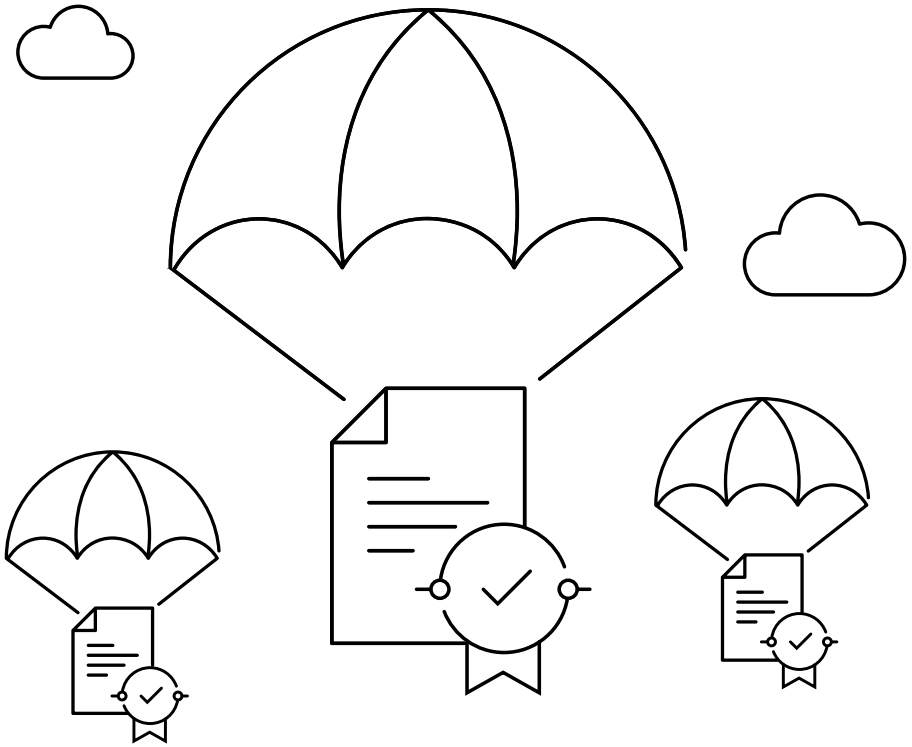
- **Seven Tips to Get the Most out of YourDevOpsPlatform**：チームが DevOps プラットフォームを最大限に活用できるようにサポートする、GitLab の eBook です。
- **Continuous Delivery**：ある読者は、開発とデリバリーのプロセス全体を結びつけることに取り組んでいるすべての人にとって、この本は「必読書」であるとレビューしています。
- **PracticalDevOps**：DevOps のしくみ、コードの保存、コードのテスト、デプロイメントなどのトピックを取り上げています。
- **TheDevOpsHandbook** は、この分野に携わる人にとっての必携書と見なされています。DevOps のメリットだけでなく、企業が競争の優位性を確保するために何をするべきかを解説しています。
- **GitLab Quick Start Guide**：GitLab プラットフォームへの移行方法がわかりやすく解説された優れたガイドです。
- **Big Little Book on Git**：経験豊富な DevOps 専門家と初心者の両方が読むべき eBook です。
- **Ten Steps Every CISO Should Take to Secure Next Generation Software**：セキュリティ担当者が、ソフトウェア開発の変化がセキュリティプログラムに与える影響を理解するための入門書です。

認定資格

- **DevOps Institute** は、開発、DevOps エンジニアリング、DevOps テスト、セキュリティエンジニアリングなどの分野で認定資格を付与しています。

- **GitLab** には、CI/CD、プロジェクト管理、DevOps セキュリティなどの分野で独自の認定資格があります。

- 情報収集はつねに源泉で。Google Cloud の使用に関するトレーニングについては、**Google ウェブサイトの認定資格ページ**をご覧ください。



ブートキャンプおよびコース

- 月額料金で、オンライントレーニングプラットフォーム **A Cloud Guru** がクラウド認定資格を提供しています。ユーザーにはビデオコンテンツ、実習、学習ツール、クイズ、テストなどが提供されます。

- **LinkedIn LearningDevOpsFoundationss** は、DevOp の初心者、またはかなりの初心者のための強力な知識ベースを提供しています。LinkedIn のサブスクリプションをお持ちの方なら誰でも無料でアクセスできます。ビデオでは、自動化、コラボレーション、モニタリング、文化などの主要な原則や技術とともに、業界の概要を説明しています。

- **TheDevOpsImplementation Boot Camp**：コンサルティング会社 Cprime が提供するこのブートキャンプでは、1,695 ドルから 3 日間のコースが受けられます。トレーニングは対面またはオンラインで行われます。プライベートチームでのトレーニングも可能です。

- **DevOps Culture and Mindset**：カリフォルニア大学デービス校が、オンラインコースプロバイダーの Coursera を通じて運営しています。完全オンラインで行われる約 15 時間のコースは、DevOps の基礎原則に焦点を当てています。Coursera のサブスクリプションに登録すれば、無料で利用できます。

- **Continuous Delivery &DevOps**：バージニア大学が Coursera を通じて提供している、初級レベルの 8 時間のオンラインコースです。継続的デリバリー、テスト、Infrastructure as Code に重点を置いています。Coursera のサブスクリプションに登録すれば、無料で利用できます。

GitLab について

GitLab は、ソフトウェアイノベーションのための最も包括的な AI 搭載の DevSecOps プラットフォームです。GitLab は単一のインターフェース、データ保管、権限モデル、バリューストリーム、レポートセット、コードの保管場所、あらゆるクラウドへのデプロイ向けの拠点、および誰もが貢献できる場所を提供します。DevSecOps の機能すべてを 1 つに統合する、真に Cloud Agnostic でエンドツーエンドを達成した、唯一の DevSecOps プラットフォームです。

GitLab を使用すると、組織はコードを迅速かつ継続的に作成、提供、管理して、ビジネス構想を具体化できます。GitLab を使うと、顧客とユーザーはより迅速にイノベーションを起こし、スケーリングもより簡単にでき、より効果的にサービスを提供して、顧客を確保できます。オープンソースで構築された GitLab では数千人の開発者と数百万人のユーザーで構成されるコミュニティとともに、新たなイノベーションを継続的に提供しています。


GitLab