シングルアプリケーションによる CI/CDのメリット

GitLab

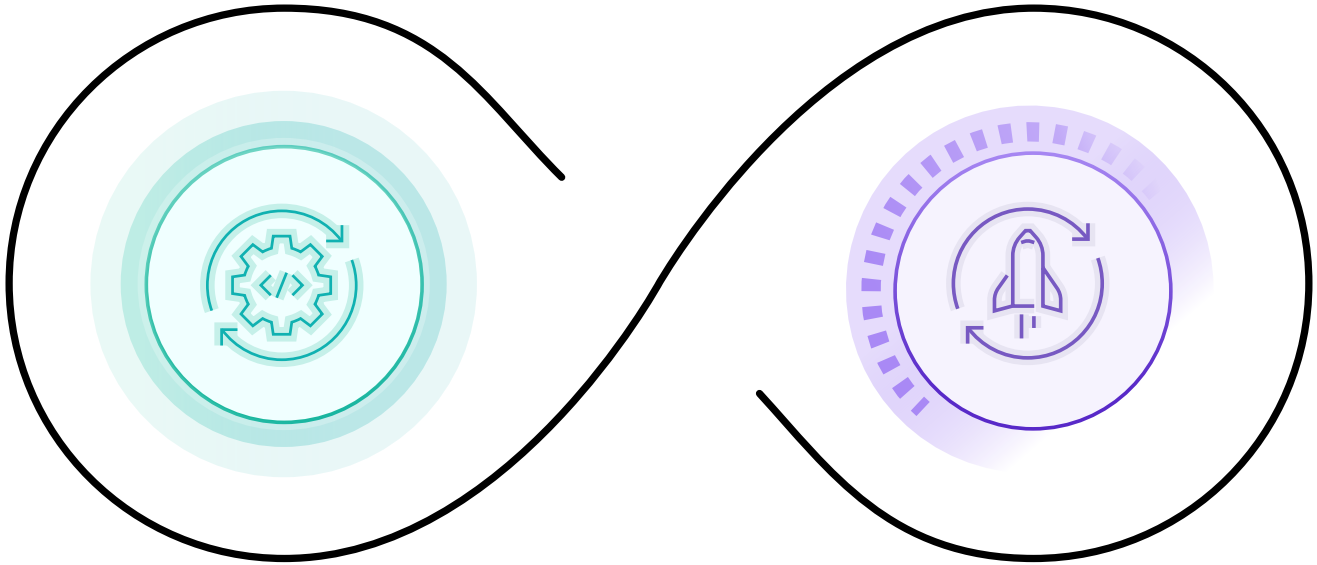
継続的インテグレーションとデリバリー（CI/CD）の利用により、ソフトウェアのビルドやテスト、デプロイへのアプローチが変わりました。CI/CDツールはこれらのプロセスを自動化することで、エラーが起こる確率を減らし、ワークフローを最適化します。各開発段階を行き来するコードとプロセス全体を通して自動化されたテストにより、エラーを確実に検出でき、本番環境に移行する前にロールバックできます。

CI/CDツールを使用する企業は増え続けており、ソフトウェア開発の方法も改善されています。CI/CDにより、年、四半期、月単位に行っていたDevOpsチームによるデプロイは、1日に数回できるようにまでなりました。

CI/CDがもたらす10のメリット

CI/CDの概要を理解したところで、そのメリットをまとめてみましょう。

1. **ハンドオフを減らせる**
   開発パイプラインでハンドオフが多ければ多いほど、障害が起こる機会が増え、プロセスがより複雑になります。



2. 開発をスピードアップする

CI/CDにより、すべての開発段階を短縮できるようになります。プロセス全体でイテレーションが高速化されるため、すべてのチームの効率が上がり、開発者は自信を持って他のプロジェクトに取り掛かれます。

3. デプロイを増やせる。

多くとも数週間に1回だったリリースを1日に6回以上行えます。

4. テストをスピードアップできる

開発ワークフローで最も時間のかかる作業が取り除かれるため、開発者は価値の高いプロジェクトに取り掛かれます。自動化されたテストにより、チームはすぐにフィードバックを得られ、本番環境（最悪の場合には最終リリース）でバグを検出するよりも早く障害に気づけます。

5. バグを減らせる

開発プロセス全体でテストが自動化されることにより、バグが発生した時点で検出でき、マスターブランチに移行する前にロールバックできます。これにより、全体のコードの質が上がり、各リリースを意図したとおりに動作させられます。

6. コンプライアンスを改善できる

コンプライアンスタスクを開発ライフサイクルに組み込むことで、コンプライアンス違反のアプリケーションをリリースするリスクを減らせます。コンプライアンスの自動化により、監査を簡単に完了できるようになり、（特に規制が厳しい業界で）コストのかかるミスを防げます。

7. イノベーション実現に多くの時間を費やせる

インテグレーションのメンテナンスに費やす時間を短縮し、差別化につながらないIT投資への資金を節約すれば、他の分野にリソースを使えるようになります。

8. 開発者が気持ちよく仕事に臨める

開発者が自信を持って作業を行い、バグを数週間後に見つけるのではなく、すぐに修正できるようになります。

9. 人件費を削減できる

組織が維持できる開発者の数には限りがあります。開発時間はかかった時間で支払いを請求されることが多く、手動でのデプロイやテストを行うとIT予算を超過する場合があります。ワークフローの自動化により、手動タスクが減り、予算を効率的に使うことができます。

10. プロセスの一貫性を保持できる

開発ワークフローでの自動プロセスを増やすことで、プロセスのステップを忘れてしまう事態を回避できます。一貫したビルドを実現できる上に、新人開発者のトレーニングを簡略化でき、組織はビルドの方法やリリースのタイミングを厳格に管理できるようになります。

CI/CDの最適化を実施

生産スピードを正確に測定するには、部分ごとではなく、SDLC全体を評価する必要があります。テック企業の役員であるGary Gruver氏は著書『Starting and Scaling DevOps in the Enterprise（企業でDevOpsを開始し、スケーリングする方法）』の中で、
DevOpsを開始する際に、開発者だけでなく、最初から新しい機能に影響を与えるビジネスアイデアまで視点を広げるアプローチを推奨しています。アイデア、開発、テスト環境、本番環境に至るまで、SDLCを開発者目線で分析しましょう。アイデアを実践するために個人が行うべきステップをすべて明らかにするべきです。このプロセスにより、組織での価値の流れが決まり、システムのボトルネックを特定できます。

シームレスな継続的インテグレーションとデリバリーを実施するには、アプリをデプロイする上で必要なツールをすべて用意しなければなりません。しかし、残念ながら、CI/CDを利用する多くの組織が、現在使用しているツールが原因で、ワークフローの最適化を阻まれています。この傾向は、何らかの問題が発生した場合に特に顕著になります。

パイプラインに障害があると、開発者は特定のツールを利用できなくなり、問題を解決するために必要な可視性を失うおそれがあります。この場合、問題を解決する権限を持つチームと問題を確認する権限を持つチームを作ってお互いに調整しなければならない場合があります。両方のチームが権限を持っている場合でも、異なるツールを使用して共同作業を進めます。たとえば、JenkinsのCIにはコードレビュー機能がないので、チームが正しい記録を行うには、SCMとCIツールを行き来しなければならなくなります。

複雑なツールチェーンにはインテグレーションと定期的なメンテナンスが欠かせません。サービスと設定が安定していない場合、中断を強いられ、ライフサイクル全体に影響が及ぶ可能性があります。その結果、ビルドの開始と完了までに待機による長期のタイムロスが発生するおそれがあります。管理者がよかれと思って「単純」なプラグインのアップデートやインストールを理由にシステムを停止することは珍しくありません。Jenkinsのクリエイターである川口耕介氏はブログ記事の中で、信頼性の欠如や使用するプラグインの多さから、Jenkinsがフランケンシュタインプラットフォームと評価されている点を含め、同プラットフォームの欠点を認めています。清々しくも正直な見解であり、Jenkinsが積極的に問題に取り組んでいる証拠ではありますが、この見解はのDevOpsチームが把握している問題を浮き彫りにしています。

それは、インテグレーションはスムーズに動作すれば問題ありませんが、停止すると地獄を見るということです。

シングルアプリケーションによるCI/CD

GitLabは、SDLC全体に可視性をもたらすシングルアプリケーションについて、すべての開発段階が網羅され、最適化されていることを裏付ける上で最高のアプローチと位置付けています。すべてを一元管理できれば、ワークフローのボトルネックを特定し、各要素がデプロイスピードに与える影響も簡単に判断できます。

単一のDevOpsプラットフォームの各ステップから、コードをマージできるかどうかという意思決定を下すために必要な情報を少しずつ得られます。オールインワンのCI/CDソリューションがなくとも、全段階を実行できますが、総合的なツールを持つことで、信頼できる唯一の情報源を設け、各ステップを素早く監視し、エラーを見逃すリスクを軽減できます。Forrester社のリサーチでも、追加設定の不要なツールチェーンソリューションにより、全体的なセキュリティが改善したとIT担当者が認めていることが明らかになっています。

継続的デリバリーはCIと緊密に連携しており、すべてのコードのテストが完了し、本番環境に移行する準備ができた際の移行プロセスもシームレスになります。効率的なCI/CD戦略は一部ではなく、最初から最後まで
自動化を採用しています。

シングルアプリケーションによるCI/CDのメリット

- 開発のライフサイクル全体を可視化できる
- すべての開発段階で信頼できる唯一の情報源（Single Source of Truth）を提供できる
- さまざまなアプリケーションへのログインが不要
- 単一のインターフェイスで操作が簡単になる
- 1つのアプリケーションのみでインストール、メンテナンス、スケーリング、バックアップ、ネットワーキング、セキュリティを実行できる
- 認証管理が簡単になる
- 運用コストを削減できる

各種CI/CDツールの違い

CI/CDツールの選択肢は豊富にあります。ソリューションの価値に差をつける大きな要素は、コストと機能です。また、現在および今後のニーズを見据えた上でCI/CDツールがふさわしいかどうかといった要素も見過ごせません。特定のプラットフォームの機能を並べて比較する方法も有効です。

GitLab CI/CDとJenkins CIの比較

Jenkinsはビルド自動化ツール兼CI/CD開発者ツールとして有数の人気を誇ります。利用可能なプラグインは数百点に達します。Jenkinsにはそれらのプラグインの機能を組み込める柔軟性があり、ビルドやデプロイ、自動化プロジェクトをサポートできます。

ビルトインCI/CD

GitLabはPrometheusを利用してデプロイされたアプリのパフォーマンスのメトリクスを収集し、表示します。開発者はGitLabを離れることなくマージの影響を判断したり、本番システムをモニタリングできます。

JenkinsもGitLabも個別のインストールが不要のCI/CDツールを用意しています。

CI/CD水平オートスケーリング

GitLab CI/CDのクラウドネイティブアーキテクチャでは、ワークロードが増えても、新しいノードを追加することで簡単に水平オートスケーリングを実行できます。GitLab Runnerは新しいコンテナを自動でスピンアップまたはスピンダウンすることで、パイプラインを直ちに処理してコストを最小限に抑えます。

クラウドコストを最小限に抑え、パイプラインを効率よく稼働させ続ける自動スケーリングランナーを提供するのはGitLabのみです。

コンテキスト依存テストの結果

GitLab CI/CDは関連するマージリクエストの中であらゆるテスト結果（インテグレーションやセキュリティなど）を返すため、開発者は諸問題につながったコード変更とその理由を特定できます。

CI/CDの結果をマージリクエスト内で提示できるのはGitLabのみです。これは、GitLabが1つのアプリケーションでCI/CDとSCMを併用しているためです。

CI/CDコンポーネントのカタログ

GitLabはパイプラインを構成するユニットを集め、リポジトリに保管するため、別のプロジェクトでパイプラインを構築する際に簡単に再利用できます。

同じアプリケーション内でカタログを提供しているのはGitLabだけです。

CI/CDログのステップフォールディング

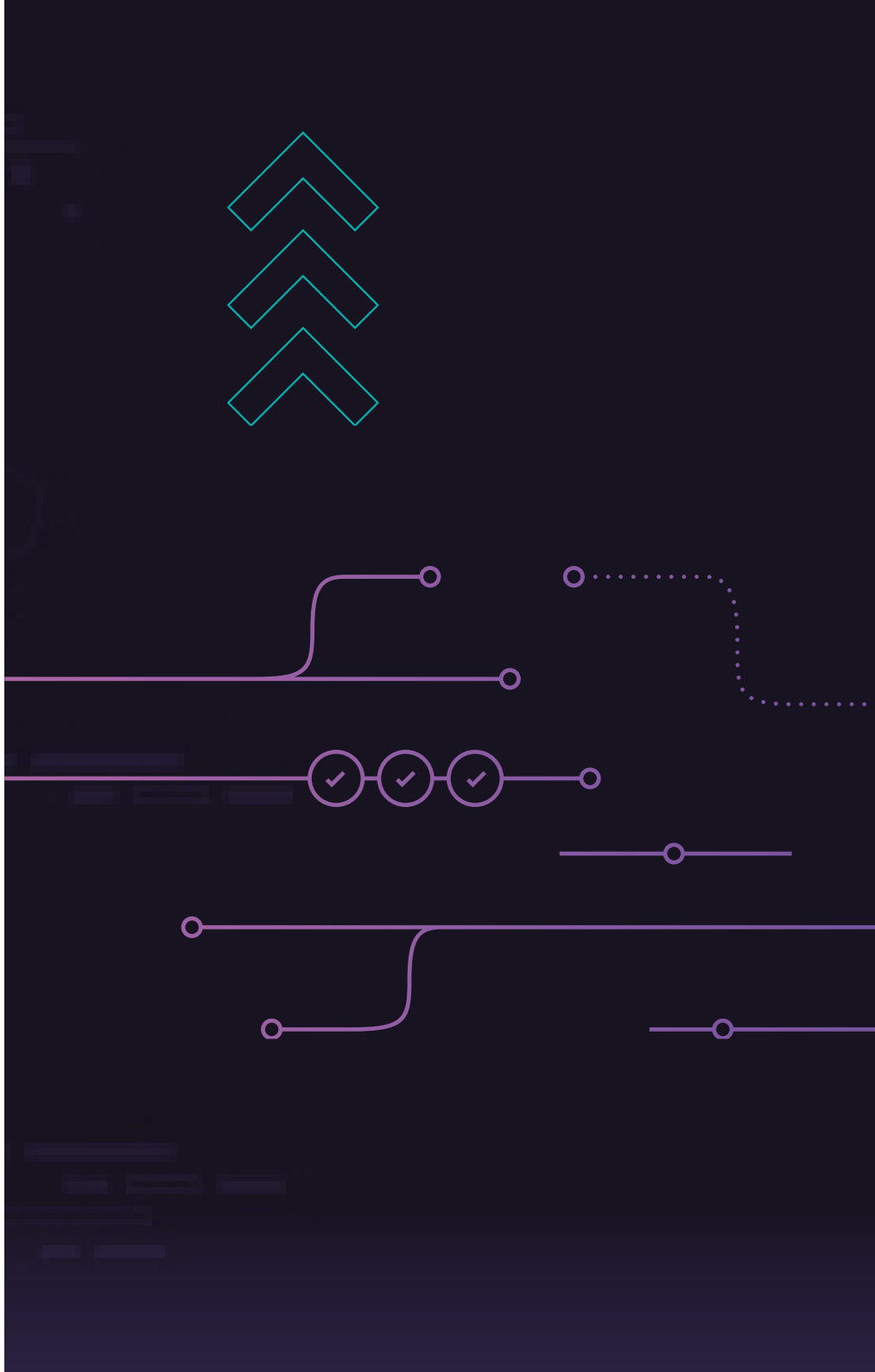
各コマンドのジョブログ出力を折りたたみます。

現時点でジョブログを折りたためるのはJenkinsのみです。GitLabは今後のリリースでこの機能を追加する予定です。

透明性は当社のコアバリューの1つであり、GitLabとその他のDevOpsツールとの比較を当社ウェブサイトで表示しているのもそのためです。Jenkinsも CI/CDを提供していますが、他のビルトイン機能の面で劣っています。
また、プラグインに依存しているため、所有コストも徐々に高くなっていく可能性があります。詳しく見る

面倒な設定は不要 そのままCI/CDでプロジェクトを開始

GitLab CI/CDはソースコードの管理や計画、モニタリングなどを含むアプリケーションにすでに組み込まれています。DevOpsのライフサイクル全体を対象としたシングルアプリケーションであるため、すべての操作を1つの会話でやり取りでき、チーム全体が内容を確認できます。GitLabのCI/CDでは追加設定は要りません。当社がCI/CDを提供する目的はDevOpsチームが管理にかける時間を短縮し、クリエイティブな作業にかける時間を増やすことです。



GitLabのCI/CDへ移行したチームの例

Ticketmaster社

GitLab CI/CDの利用で週1回のモバイルリリースが可能に

「2月以降、当社ではモバイルアプリを毎週リリースしています。過去数回のリリースでは、GitLab CIが成功に大きく貢献していました。サイクルタイムの短縮やリリースの早期化のほか、さまざまなメリットを実感しています。各リリースの変更セットは小規模になり、クラッシュフリー率とストアでの評価が向上しました。また、ビルドの待機時間が短くなったことで、製品の品質向上に費やせる時間が増えました。Ticketmasterのお客様は、新機能をより早く入手できるようになったほか、高品質の製品が継続的に改善されるようになったことで、恩恵を得ています。GitLabのCI分析を介して、チームが将来に向けた最適化と改善を行う上で有益な情報を得られます。」

— Android開発チームリードエンジニア
Jeff Kelsey氏

同社の事例を読む

HackerOne社

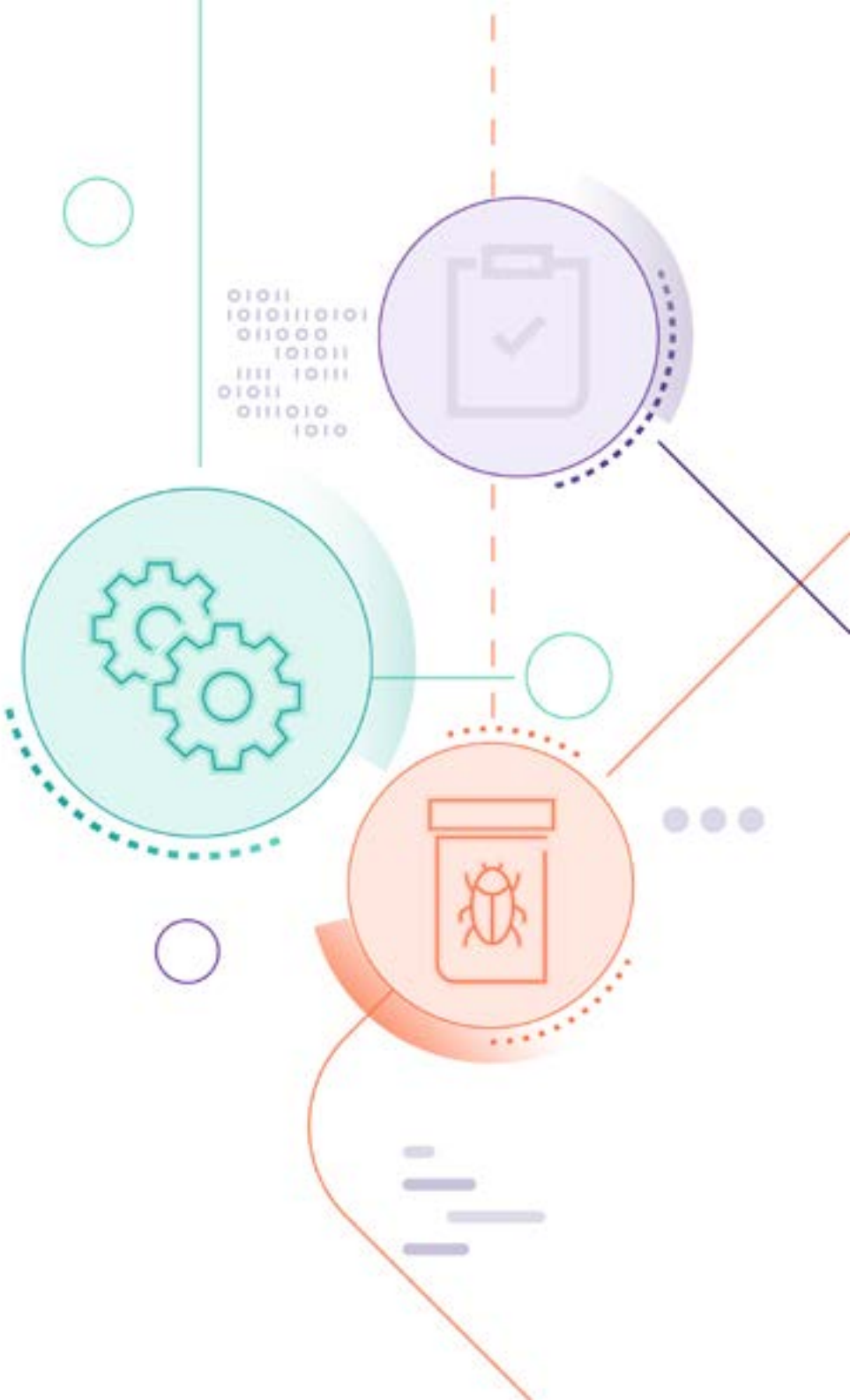
CI/CDの一元化がサイクルタイムの短縮やパイプラインの可視性の向上、セキュリティスキャンの前倒しを実現

「パイプライン管理の民主化に成功しました。当社ではエンジニアがそれぞれDevOpsとしての役割、インフラとしての役割を果たしています。また、今までできなかった方法でツールの構成を管理できるようになりました。以前のツールは扱いにくく、メンテナンスと管理が大変でした」とTrale氏は述べています。さらに同氏は「GitLabではセキュリティ上の欠陥を早期発見でき、これは開発者のフローに組み込まれています。エンジニアはコードをGitLab CIにプッシュし、数ある段階的な監査ステップの1つから即座にフィードバックを得て、そこにセキュリティの脆弱性が組み込まれているかどうかを確認できます。さらに、特定のセキュリティの問題をテストするための追加のステップを構築できます。」と続けました。

— HackerOne社 インフラストラクチャ部門長
Mitch Trale氏

同社の事例を読む

CI/CDツールにはプロセスを自動化し、コードの質を高められる効果があり、組織の時間と資金の節約に貢献します。コードの自己テストやビルドの自動化、バグの早期検出、進捗のモニタリングといった機能により、時間効率を最適化してリリースサイクルを短縮できます。また、CI/CDの競争優位性は、アプリケーションにとどまらず、チーム間のハンドオフを促進し、一体感のあるDevOps文化を創生することにあります。プロセス効率が上がり、手動タスクが減り、可視性が上がることで、定着率も高まり、より優秀な人材を惹きつけられます。シングルアプリケーションによるCI/CDを使用すれば、チームはより質の高い、シームレスな体験を目指して、こういったメリットをさらに拡大していけます。



GitLabの無料トライアルを開始

GitLabについて

GitLabはシングルアプリケーションの総合DevOpsプラットフォームです。Concurrent DevOpsを可能にするのはGitLabだけです。現代のツールチェーンがもたらす制約から組織を開放できます。GitLabは可視性に優れ、桁違いの効率性と総合的なガバナンスを誇り、変更の計画からその効果のモニタリングまでの時間を大幅に短縮します。その結果、ソフトウェアのライフサイクルは200%速くなり、業務のスピードを大幅に改善します。

GitLabとConcurrent DevOpsはソフトウェア開発ライフサイクルの全段階を通して効率性アップを達成し、サイクルタイム短縮を実現します。製品、開発、品質管理、セキュリティ、運用チームがシングルアプリケーションで初めて同時に作業を進められます。ツールを統合して同期したり、ハンドオフを待って時間を無駄に費やす事態を防げます。1つの会話に全員が参加でき、さまざまなツールにまたがる複数のスレッドを管理する手間を省けます。また、単一の信頼できる情報源を介してライフサイクルを完全に可視化し、トラブルシューティングを簡略化して管理記録を強化できるのはGitLabだけです。すべてのアクティビティを一貫したコントロールの下に管理し、セキュリティとコンプライアンスを後付けではなく、最優先事項として扱います。

オープンソースで構築されたGitLabは、数千人の開発者と数百万人のユーザーから成るコミュニティの貢献により、新たなDevOpsイノベーションを次々に実現しています。Ticketmaster社、ING社、NASDAQ、Alibaba社、Sony社、Intel社など、10万社を超える企業が、優れたソフトウェアをスピーディーに開発するGitLabの能力を信頼しています。


GitLab