ミッションの迅速な遂行：スピードの戦略的な価値

GitLab

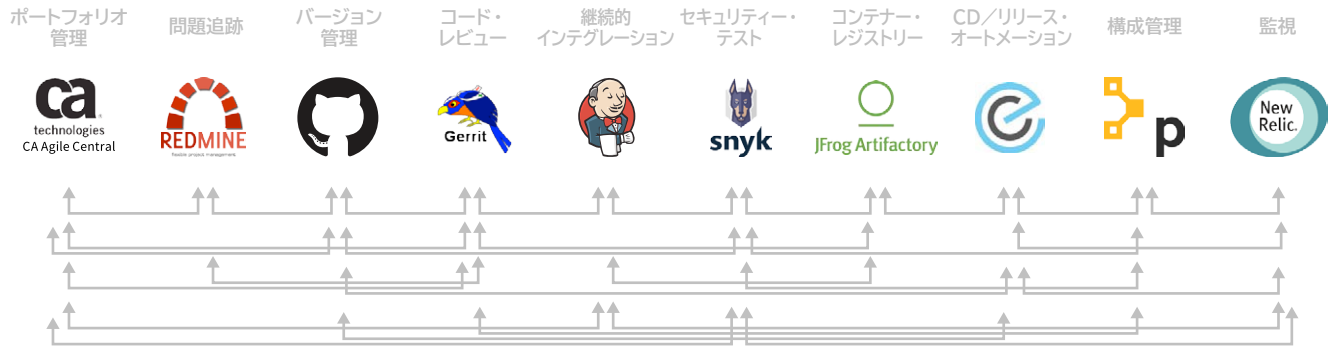
優先課題の急速な変化、新たな脅威の出現、デジタル・サービスに対する市民の期待の高まりという現状にあって、あらゆるレベルの政府機関が、デジタル・アプリケーションやシステムを合理化するための新たな方法を見つける必要に迫られています。詳細な要件の膨大なリストを定義し、請負業者またはシステム・インテグレーターがソリューションを提供してくれるまで何年も待つという従来のアプローチは、もはや実用的な選択肢ではありません。経済においても、また私たちの身近でも、「戦いの場」における変化のペースはますます高まり、今や、スピードと迅速な対応が新たな規範として求められるまでになっています。

アジャイル・プランニングやDevSecOpsデリバリーの実証済みの手法によって、アプリケーション・デリバリーを合理化し、高速化する、堅牢性、信頼性、スケーラビリティーに優れたソリューションが実際に提示されています。自動インテグレーション、テスト、デジタル・アセットの管理、脆弱性のスキャン、アプリケーションの反復的なデプロイと構成を可能にする新たなツール群を活用することで、開発チームは、実用的なアプリケーションのビルド、テスト、デリバリーに必要な、相互に依存するツールと作業のチェーンを構築できるようになりました。

ツールチェーン

このような統合されたツールチェーンは、アプリケーション・デリバリーの高速化に役立つ反面、複雑性、データ・アイランド、一貫性のないセキュリティー設定、レポートにおける難題、コンプライアンス上の問題という形で新たなコストとオーバーヘッドをもたらします。新たなツールが追加されるたびに、新たなインテグレーションが必要となり、アプリケーション・デリバリー・チームの作業全体に複雑さをもたらす新たな要因が加わることになります。プロジェクト管理者、開発者、テスト担当者、運用チーム、セキュリティ・チームが、それぞれ独自のツールを導入することで、全体の複雑さが増します。

ツールチェーン・インテグレーションに伴う問題点



開発チームが実際に手にしているのは、複雑で、壊れやすく、高コストの、フランケンシュタインのようなツールチェーンです。こうしたツールチェーンでは、価値を提供するどころか、アセンブリー・ラインのツールの管理に伴う無駄な作業にサイクル・タイムが浪費されるはめになります。このDevOpsツールチェーンの管理オーバーヘッドは、「ツールチェーン・タックス（ツールチェーンに伴う負担）」のようなものです。開発チームが必要としているのは、クリーンでモダンなソフトウェア・ファクトリーです。つまり、数十もの異種のツールやカスタム・インテグレーションの管理における無駄やオーバーヘッドを伴うことなく、アプリケーションのビルド、テスト、デリバリーを迅速に行える、効率的で管理しやすい、十全に機能するアセンブリー・ラインを備えたソフトウェア・ファクトリーなのです。

このソフトウェア・ファクトリーは、以下の課題に対処できるものでなければなりません。

1. 問題の特定と計画

デリバリー・チームは、新たな要件やユース・ケースの取り込み、議論、優先順位付け、定義を行うことができなければなりません。新たな問題が、エンドユーザーが求める特定の機能のユース・ケースと要件をもたらします。製品管理者、デリバリー・チーム、および顧客によって、ユース・ケースが定義され、ニーズについて詳述され、重要度に基づいて新たな機能の優先順位が付けられます。問題とユース・ケースは、事実上、ファクトリーの注文入力プロセスであり、これらによって将来の開発が開始されます。

2. 分散ソース・コード管理

アプリケーションの設計と開発は、ソース・コード内の分岐の管理、複数のファイルの頻繁な変更の追跡、そうした変更により脆弱性が生じないようにするための保護、メイン・トランクへの変更のマージとインテグレーションを必要とする集中的なアクティビティーです。分散ソース管理を通じて、開発者は、中央のリポジトリーから独立して作業しながら、より大きなチームとの間で作業を同期させることができます。分散ソース・コード管理により、ソフトウェア開発チーム全体での調整、共有、コラボレーションが可能になります。

3. コードのレビューと承認

ピア・レビューと厳格な承認は、新たなコード変更によりユーザーのニーズに確実に対処し、論理エラー、欠陥、セキュリティーの脆弱性をもたらさないようにするうえで不可欠です。通常、コードの変更を承認するには、変更について明確に文書化し、追跡して、コンプライアンスを実証する必要があります。この重要な監視およびレビュー・プロセスをソフトウェア・ファクトリーのコア機能とし、品質、説明責任、コンプライアンスを徹底すべきです。

4. すべてのコミットでの継続的インテグレーション（CI）

ソフトウェア・ファクトリーのバックボーンは、各コード変更での開発作業の完了を自動化する継続的インテグレーション・パイプラインです。CIパイプラインにより、正しい順序での、自動化されたテスト、スキャン、コンプライアンス・チェックの完了が徹底されます。

a. ソフトウェアの品質（自動テスト）

CIパイプラインにより、すべてのコミットに対して、単体テストから、APIテスト、機能テスト、非機能テストまで、さまざまな自動テストが管理されます。継続的インテグレーション・パイプラインにおける自動テストの目標は、テストを高速化することと、新たなコード変更によって新たな欠陥や問題が発生しないようにすることです。

b. セキュリティー（コードのスキャン、コンテナーのスキャン、ライセンス管理、依存関係のスキャン）

アプリケーション・セキュリティー・スキャンをCIパイプラインに組み込んで、新たな脆弱性や問題をもたらすソフトウェアの変更についてのフィードバックを迅速に得られるようにすべきです。開発ライフサイクルのできるだけ早い段階でセキュリティー・フィードバックを得て、セキュリティーの問題が原因でソフトウェア・デリバリーが遅れるリスクを最小限に抑えることがきわめて重要です。要件から、コーディング、テスト、デリバリーまで、デリバリー・プロセス全体でセキュリティーの発想を浸透させるべきです。目標はDevSecOpsです。

5. バイナリー・アセットを管理するレポジトリー

継続的インテグレーション・パイプラインのアウトプットは、アプリケーションを構成するバイナリー・コードとライブラリーです。これらのアセットを管理し、アプリケーションのテスト、検証、デプロイのすべての過程で追跡する必要があります。

6. 継続的デリバリー（CD）

継続的デリバリー・パイプラインとは、開発されたアプリケーションを必要な環境にデプロイして構成できるようパッケージ化するために必要な、自動化された作業とステップです。CDパイプラインはCIパイプラインをそのまま拡張したものである場合があり、このパイプラインによって、開発から、デプロイ、実稼働の準備、そして最終的には実稼働へと、作業のアセンブリー・ラインがつながれます。

7. 動的テスト環境／インフラストラクチャー

開発作業を合理化するために、ソフトウェア・ファクトリーでは、要求に応じてデプロイ可能で、個々の開発者やチームのテストのニーズに対応できる、動的な（一時）テスト環境をサポートすべきです。従来の方法では、新たなコード変更があっても、テスト環境やリソースが限られているために、それらを使用できるようになるまで待ってからテストを行うことになります。ファクトリーでは、コンテナー化およびクラウド・テクノロジーを活用し、テスト環境が使用可能になるまで待っている間に発生する遅延を軽減するか、なくさなければなりません。

8. 漸進的なデプロイ

ファクトリーからのソフトウェアのデプロイでは、漸進的なデプロイをサポートして、チームがリスクを最小限に抑えられるようにする必要があります。ソフトウェア開発チームは、カナリア・デプロイや機能フラグなどの手法を用いて、リスクを積極的に管理・軽減しながらコードを迅速に出荷するための柔軟性を確保することができます。

9. アプリケーションの監視

実稼働中のアプリケーションからのフィードバックは、モダン・ソフトウェア・ファクトリーにとって不可欠です。アプリケーションの監視を通じて迅速に得られる実用的な洞察は、製品開発者が問題を検出し、必要な措置を講じて、アプリケーションの継続的な向上を図るために役立ちます。

モダン・ソフトウェア・ファクトリーは、アプリケーションのビルドとデリバリーの迅速な実施という課題に対処するために必要なコラボレーション、共有、チームワークを促進します。

GitLabは、開発チームがミッション・クリティカルな機能をより短期間でデリバリーできるようにするための、包括的なソフトウェア・ファクトリー・ソリューションです。オープンな、拡張性を備えたGitLabは、要求のきわめて厳しい組織の独自の要件にも柔軟に対応することができます。GitLabは、単一のアプリケーションでありながら、ソフトウェア・デリバリー・ライフサイクル全体のエンドツーエンドの要件に対処できます。

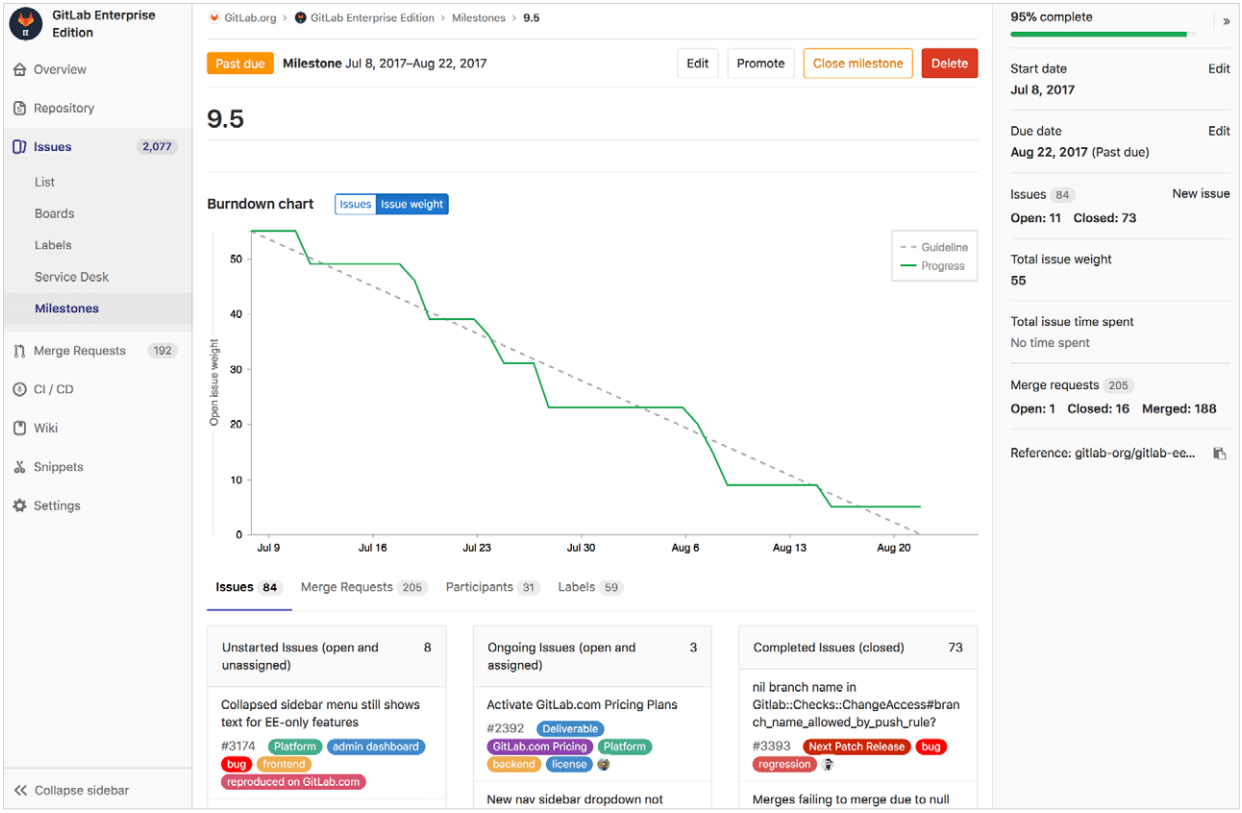
単一のアプリケーションであるGitLabは、次のような独自の価値をデリバリー・チームにもたらします。

- ツールチェーン全体での、単一の共通ユーザー・エクスペリエンス
- 共通のセキュリティーおよびアクセス・モデル
- 信頼できる唯一の情報源に基づく、開発作業のレポートと管理
- コンプライアンスと監査の簡素化
- 請負業者や管理職から、エンドユーザー、開発者まで、すべての人が1つの会話に加わり、貢献
- 統一されたガバナンス・モデル

完全なソフトウェア・ファクトリーを構築するGitLabの個別機能

- **アジャイル・プロジェクト管理**（エピック、問題、カンバン・ボード）

  エピック、問題、カンバン・ボードを使って作業に優先順位を付けて構造化することにより、将来のソフトウェア・プロジェクトと作業の計画を容易に管理することができます。開発チームは、作業を整理して、時間ベースのスプリントを計画したり、ビジネスの要求に基づいて新たな機能の開発とデリバリーを継続的に行うリーン・フロー原則に従って作業の計画を立てたりすることができます。



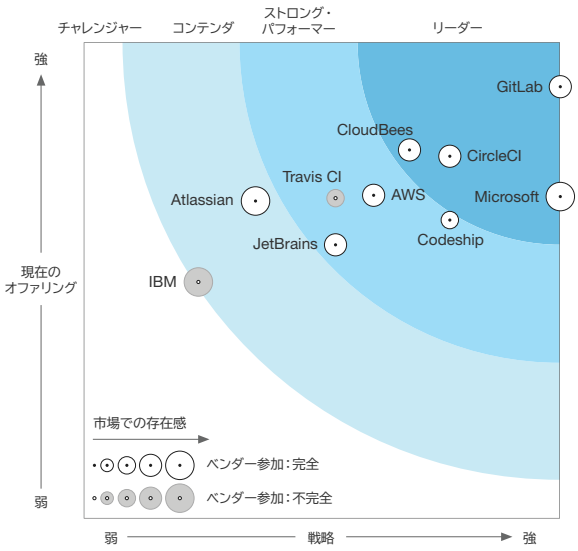
» Gitに基づく分散バージョン管理

GitLabマージ・リクエストを使って、複数のバージョンやソース・コードの変更を効率的に管理し、表示することができます。この機能により、特定のコード変更に関してチーム間でコラボレーションしたり、コードのレビュー、セキュリティー・テストの結果のレビュー、アプリケーションの変更の承認を行ったりすることができます。マージ・リクエストでは、適切な担当者によるコードのレビューおよびコード変更の承認に関するポリシーを適用することができます。Gitに基づくGitLabのソース管理システムにより、各開発者は、プロジェクトのソース・コードの分散するインスタンスに対して効率的に作業を行い、行った作業を定期的に同期させることができます。

» 業界をリードする開発オートメーション

継続的インテグレーション（CI）と継続的デリバリー（CD）

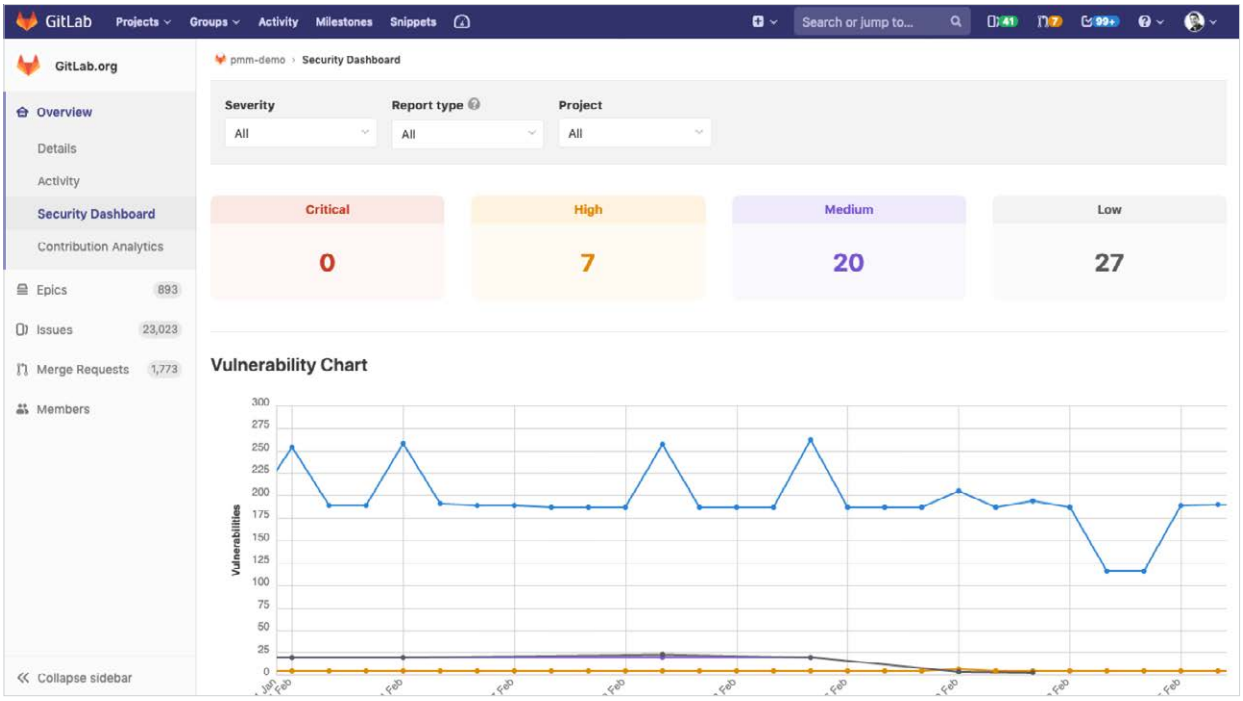
単一のコード行に対する従来のワークフローでは、更新後のアプリケーションをデプロイしたり構成したりする前に、アプリケーションのコンパイルとビルド、コード品質のスキャン、単体テスト、機能テストのための手動によるステップが必要となることがしばしばあります。GitLabでは、これらの手動作業がGitLab CI/CDパイプラインで自動的に実行されます。CIパイプラインでは、アプリケーションの変更のビルド、テスト、および認証といった作業が自動化されます。オートメーションには、並行して実行することで時間を節約できる作業を含めることができます。例えば、パイプライン・オートメーションにより、コードを変更するたびにコンプライアンス関連の特定の作業が確実に実行されるようにすることができます。CDパイプラインでは、インフラストラクチャーのプロビジョニングとアプリケーションのインストールや構成に必要なステップと作業が自動化されます。GitLabでは、CDパイプラインによって、コンテナーへのアプリケーションのパッケージ化と環境（OS、VM、またはKubernetes）へのアプリケーションのデプロイが簡素化されます。CI/CDパイプラインの基本的なメリットは、ステップの一貫した実行、サイクル・タイムの迅速化、デリバリーに大きな影響を及ぼす可能性がある手動によるミスの削減です。



Forrester CI Wave™で最高の評価を獲得

「GitLabは、適切に文書化されたインストール・プロセスと構成プロセス、分かりやすいUI、セルフ・サービスをサポートする柔軟なシート単位の価格設定モデルを通じて、開発チームをサポートしています。GitLabのビジョンは、コードの作成に時間をかけるようにして、ツールチェーンのメンテナンスに費やす時間は減らしたいと思っている、エンタープライズ規模の統合されたソフトウェア開発チームにサービスを提供することです」
— Forrester CI Wave™

» セキュリティーのスキャンとレビュー



モダン・アプリケーションの構築時には、アプリケーションのセキュリティーを第一に考える必要があります。GitLabでは、組み込みのアプリケーション・セキュリティー・スキャン機能によって、あらゆるコード変更の迅速なフィードバックが提供されます。セキュリティー・スキャンによるコードの静的評価とアプリケーションの動的評価を通じて、潜在的な脆弱性が明らかにされます。動的なセキュリティー・スキャンによって、実行中のアプリケーションがスキャンされ、潜在的なリスクがないか確認されます。フィードバックが迅速に返されることで、開発者はデリバリー前の最終段階まで待たなくても、開発プロセスの早期の段階でセキュリティー・リスクに対処することができます。GitLabは、デリバリー・ライフサイクル全体でDevSecOpsを実現し、組み込みのコンテナー・スキャン、ライセンス管理、アプリケーション・セキュリティー・ダッシュボードといった機能を通じて、開発者が安全で信頼できるコードを提供できるよう支援します。

» レビュー・アプリ

コード変更の迅速な評価によって、開発者は自分が加えた変更を検証し、追加調整を行うことができます。ただし、アドホック・テスト用の非実稼働環境をすぐには使用できず、待たされることもよくあります。GitLabのレビュー・アプリケーションは、あらゆるコード変更に使用できる実稼働前環境です。この環境を使用することで、開発者は、テスト環境などの限られたリソースが使用可能になるまで待つことなく、自分のアプリケーションをすばやく実行して評価することができます。

» カナリア・デプロイと機能フラグ

大規模な変更が加えられたアプリケーションをエンドユーザーにリリースすることは、大きなリスクを伴います。予期せぬバグや問題によって、ユーザーの生産性や、組織におけるミッションの目標達成能力に重大な影響が及ぶ可能性があります。GitLabでは、こうしたリスクの軽減に役立つ漸進的なデプロイをサポートしています。例えば、カナリア・デプロイでは、特定の変更が加えられたアプリケーションを一部のユーザーのみにデプロイすることができます。またGitLabでは、アプリケーション内の特定の機能を選択して有効または無効に設定することのできる機能フラグも提供しています。これらの機能を使用することで、チームはより小さな変更を漸進的に加えながらアプリケーションを配布して、組織全体にリスクがもたらされるのを避けることができます。

» アプリケーションの監視

複雑なアプリケーションは数十もの個別コンポーネントで構成され、多くの場合、これらのコンポーネントは相互に依存しています。機能が設計どおりに動作していないときには、開発者と生産チームに早期に警告が送信される必要があります。GitLabには、開発者やサポート・チームがアプリケーション内の問題の検出、診断、トリアージを行うのに役立つ、アプリケーションの監視およびトレース機能が用意されています。

その他の機能として、以下の機能があります。

» LDAPインテグレーションとCaCカードのサポート

IDとアクセスの管理は、どのアプリケーション開発プロジェクトでも大きな問題です。GitLabは、LDAPサーバーとのインテグレーション、およびCaCカードなどのトークンのサポートを通じて、ユーザーのIDとGitLabアプリケーションへのアクセスを管理します。

» 包括的なプロジェクトのエクスポート

機密扱いの設定が必要なアプリケーションや規制された環境で使用されるアプリケーションを開発する場合の一般的なユース・ケースの1つは、開発作業を非機密コンポーネントに細分化して、より低いセキュリティー設定で各コンポーネントを開発できるようにすることです。GitLabの強力なプロジェクト・エクスポート機能により、各チームはよりセキュリティーの低い環境で作業してから、プロジェクト・コード全体、ディスカッション履歴、要件、パイプライン、構成をエクスポートして、保護された環境内で作業を行うチームに引き渡すことができます。これにより、実質的にプロジェクトの完全なコンテキストがエクスポートされ、新たなチームに提供されます。

» 高可用性と障害復旧

アプリケーションのダウンタイムは開発者の生産性に多額の損失をもたらす可能性があります。GitLabは高可用性モードで動作するよう構成できるため、単一障害点が排除されます。またGitLabは、地理的分散モードで動作するように構成することもできます。このモードでは、複数の読み取り専用インスタンスが同期されるため、ローカルの開発者が短時間で応答できるようになるとともに、障害が発生した場合の緩和策にもなります。

運用権限（Authority to Operate：ATO）の迅速な取得

政府機関はしばしば、新たなアプリケーションや更新されたアプリケーションを迅速にデプロイするのに苦労することがあります。これは、運用権限（Authority to Operate: ATO）と呼ばれる、自機関のガバナンスの承認を得るのが難しいためです。開発チームが必要としているのは、アプリケーションのビルド、テスト、デリバリーを迅速に行える、効率的で管理しやすい、十全に機能するアセンブリー・ラインを備えた、クリーンでモダンなソフトウェア・ファクトリーです。ガバナンスのポリシーとテストは、自動化して、CI/CDのビルド、テスト、デプロイの各プロセスに組み込むことができます。このアプローチにより、ATO承認のタイムラインを大幅に短縮できます。

さらに、GitLabでは、実装の強化について検証するために必要なステップも導入しています。これらのステップを経ることによって、あらゆる種類の機関で、完全に保護された、脆弱性のない実装が保証されるようになります。商業セクターの金融機関から、国防省などの政府機関まで、さまざまな機関が、このステップを通じて、DevSecOpsライフサイクルでの高度な信頼性を確保できます。

DevOpsツール・タックスの排除

GitLabは、ソフトウェア・ファクトリーのコンポーネントの設計、開発、テスト、インテグレーション、ATO取得のプロセスを簡素化します。この簡素化によって、ツールチェーン・「タックス」（独立した複数のツールの管理オーバーヘッド）が減少し、組織はリソースをミッション・クリティカルなアプリケーションのビルドとデプロイに集中的に割り当てられるようになります。

GitLabは、ソフトウェアの開発／デリバリー・ソリューションとして、何十万もの組織に信頼されています。GitLabは、AWS、Azure、AWS GovCloud、AWS C2Sなどのクラウドにデプロイされ、民間、国防省、情報コミュニティー機関のアプリケーション開発プロジェクトをサポートしています。

モダン・ソフトウェアのアーキテクチャーおよび手法は、取得コストを抑えながら、複雑さを緩和し、価値創出を高速化できるレベルにまで成熟しています。DevSecOpsおよびアジャイル開発プラクティスを導入することで、チームは、反復的なソフトウェア開発を実践し、ミッション・クリティカルな機能要件のデリバリーに優先順位を付けることができ、またコミュニケーションと文化的規範を合理化することができます。サイバーセキュリティーおよびセキュリティー管理をソフトウェア・デリバリー・プロセスの基礎に置くことが最も重要です。私たちGitLabは、誇りを持ってこのデジタル・トランスフォーメーションの一翼を担い、さまざまな機関の皆様が、迅速にミッションを達成し、構成員のきわめて重要なニーズを満たせるよう支援しています。

GitLabについて

GitLabはDevOpsライフサイクル全体で使用可能な初の単一アプリケーションです。同時DevOpsを実現して、今日のツールチェーンの制約から組織を解放できるのは、GitLabだけです。GitLabは、比類なき可視性、抜本的な新しいレベルの効率、包括的なガバナンスを提供することによって、変更の計画から効果の監視までの時間を大幅に短縮します。これにより、ソフトウェア・ライフサイクルが200%高速化され、ビジネスのスピードが急激に高まります。

GitLabと同時DevOpsは、ソフトウェア開発ライフサイクルのすべての段階で効率を向上させることによって、サイクル・タイムを短縮します。製品、開発、品質保証、セキュリティー、運用の各チームが、単一のアプリケーションで同時に作業することがはじめて可能になりました。さまざまなツールを統合して同期させる必要も、ハンドオフを待つ必要もありません。異種のいくつものツールで複数のスレッドを管理するのではなく、すべての人が1つの会話に貢献します。また、信頼できる単一のデータ・ソースに基づいて、ライフサイクル全体の完全な可視性を提供することによって、トラブルシューティングを簡素化し、説明責任を促進できるのも、GitLabだけです。すべてのアクティビティーが一貫性のある管理によって統率され、セキュリティーとコンプライアンスは、後から考慮されるのではなく、最初から第一級オブジェクトとして扱われます。

オープン・ソースを基盤とするGitLabは、何千もの開発者と何百万ものユーザーによるコミュニティーへの貢献を活用しながら、DevOpsの新たなイノベーションを継続して提供しています。Ticketmaster、ING、NASDAQ、Alibaba、Sony、Intelをはじめとする100,000以上の組織が、GitLabによる、優れたソフトウェアのこれまでにないスピードでのデリバリーに信頼を寄せています。

無料トライアルを開始する


GitLab