8つの国内事例を掲載

DevOps事例集

株式会社NTTデータ

オーケー株式会社

株式会社カーフロンティア

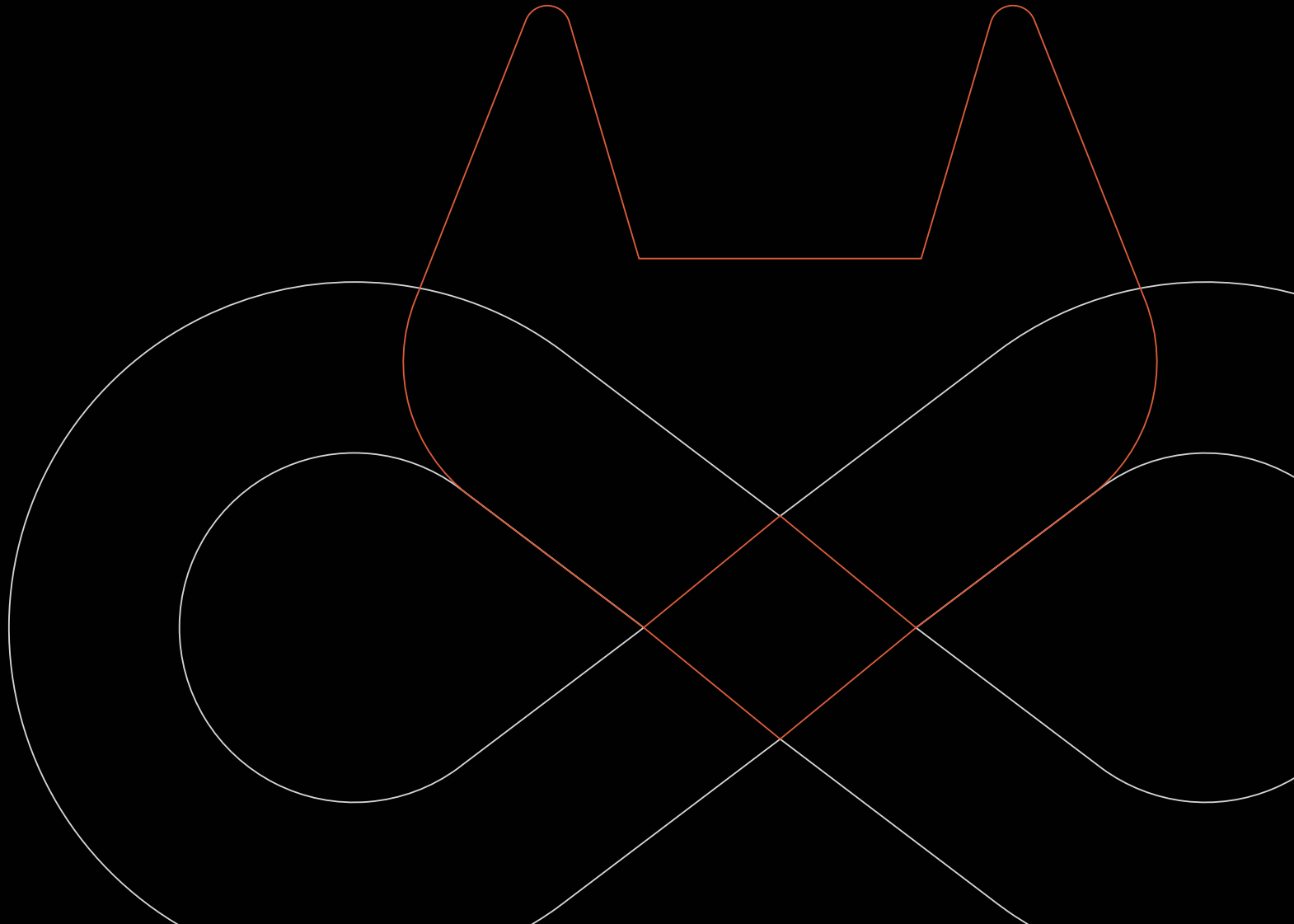
株式会社ジークス

凸版印刷株式会社

弁護士ドットコム株式会社

株式会社フィックスターズ

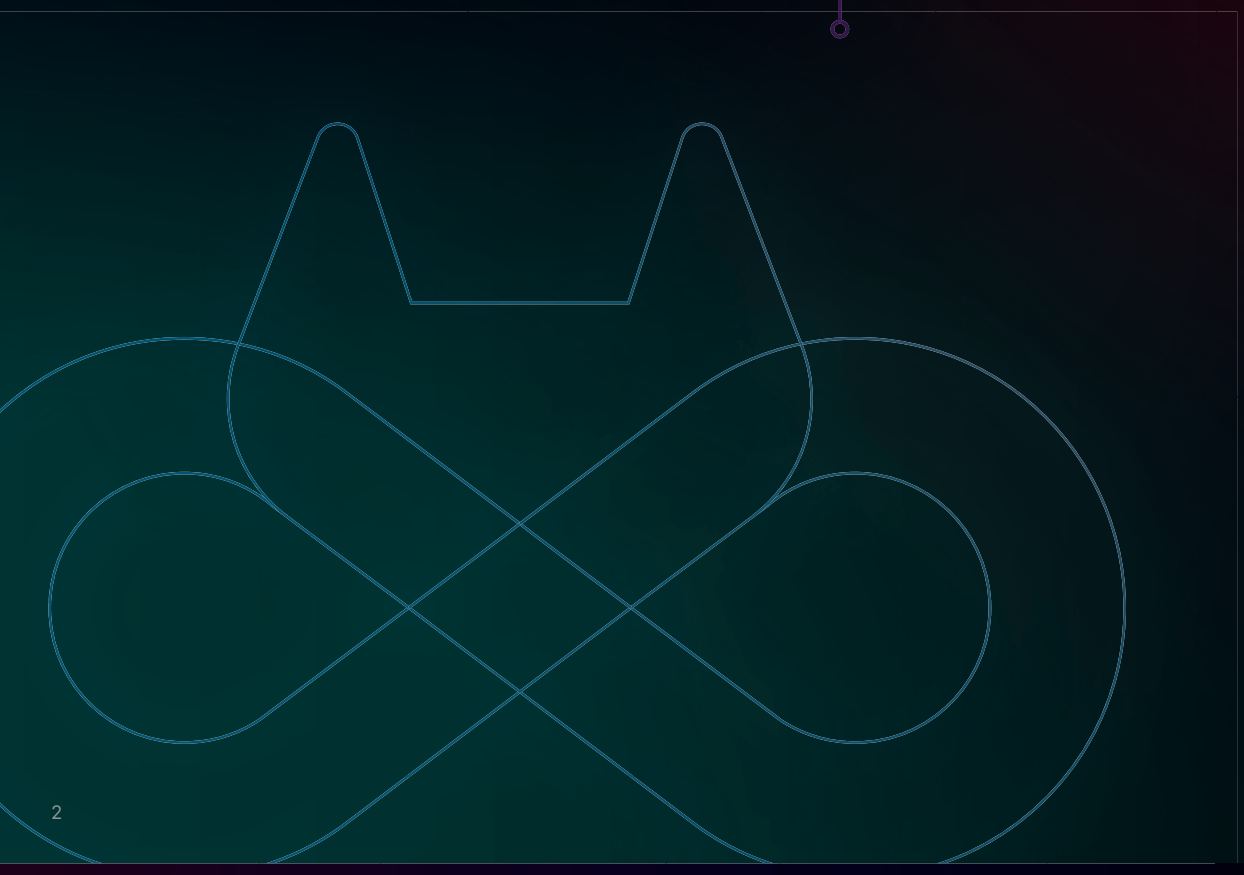
ルネサスエレクトロニクス株式会社

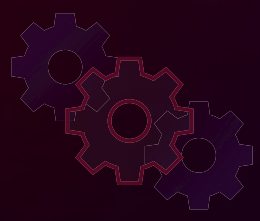


GitLab

GitLabについて

GitLabは「DevSecOpsプラットフォーム」を提供するソフトウェア企業です。
GitLabは全世界で3000万人以上に利用されており、プロジェクトマネジメント、ソースコードマネジメント、IDE、CI/CD、セキュリティ、オブザーバビリティ、AIなど、
DevSecOpsを実現するために必要なすべての機能を1つのプラットフォームに統合して提供しています。
これによりソフトウェア開発ライフサイクルにおける、プロセスの可視化や改善、情報やデータの一元管理が可能となり、開発の効率化だけでなく、
ガバナンスの強化、ディベロッパーエクスペリエンスの向上が期待できます





目次

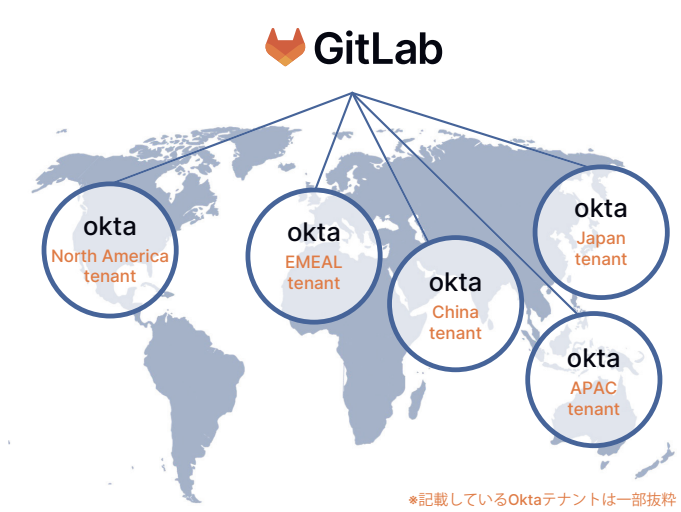
グローバルでアセット流通を目指せ！
NTTデータにおけるGitLab活用の取り組み紹介

GitLab導入の背景

従来型のSIのようなスクラッチ開発から脱却し、業界のベストプラクティスやグローバルテクノロジーをアセット化。提案・コンサルティングから開発、運用までの一連のプロセスにおいてそのアセットを活用し、ビジネスの俊敏性向上と顧客企業への提供価値の最大化を目指すものだ。
デジタル領域における更なる競争力の強化には、各地域のナレッジやアセットの集約、活用が不可欠となる。海外ビジネスをM&Aにより拡大してきた背景から、各地域のナレッジ・ノウハウ・アセットなどが散在していることが課題だった。
アセットベースのビジネスモデルを実現するには、海外含めたNTTデータのグローバル全体でアセットを共有するプラットフォームが必要となる。そこで、ソフトウェアやナレッジ、情報などを共有のプラットフォームである「アセットリポジトリー」として一元管理し、社内に展開する取り組みを2022年度から進めている。
そして、その中のソフトウェアリポジトリーにGitLabを活用することとしたのだ。

GitLabの選定理由・活用イメージ

グローバルでサービスを展開するうえでのソリューション選定の前提条件は、「グローバルスタンダードな製品であること」だ。市場シェアやサポート体制・言語などに加え、100項目以上にのぼる社内情報システムのセキュリティ対策基準への適合などを経て選定が進められた。
そして、最終的にGitLabを選定した理由は大きく3つある。1つめは「各地域の認証基盤との連携が可能」である点だ。NTTデータでは、認証基盤としてOktaを利用している。そこで各地域で独立した複数のOktaテナントとの認証連携が必要だった。



※記載しているOktaテナントは一部抜粋

その点、GitLabのSelf-managed版は、複数認証基盤の接続をサポートしており、各地域のOktaとの接続が可能であることと、さらにAWS上での動作をサポートしているため、自社データセンタでのインフラ運用は不要である点が決め手となった。
2つめは「グループやサブグループによる権限管理」だ。アセットはグローバルで広く展開する必要があり、登録したアセットを基本的にグローバルで参照可能な状態にしたいと考えた。しかし、地域特性が高いアセット、機密性が高いアセットについては公開範囲を限定する必要があり、公開範囲の制御が求められた。その点、GitLab Groupsを利用することで、公開範囲の制御が可能であることが確認された。
3つめは、「アセット共有に適した機能・ライセンス」という点だ。グローバルでのアセット共有ということで、数万人の社員がアクセスする可能性があり、アセット共有に適したライセンス・機能が必要となる。さらに品質やセキュリティ確保の観点からは、グローバルで公開するアセットのため、特に、セキュリティの脆弱性は公開前に見つけたいという要望があった。
こうした要件に対しても、GitLabの「Ultimate License」を活用することで、コミット時におけるセキュリティチェック機能により、脆弱性のあるアセットがグローバルに展開されることを防ぐことが可能。また、無料のGuest userを無制限に作成可能であることも相まって導入が決定した。
現在、NTTデータでは、開発用と公開用の2つのGitLabを活用している。
アセット開発用GitLabは、それまで海外拠点で利用していた既存のGitLabを活用している。そこで、各地域の認証基盤であるOktaとの認証連携を行う必要があった。GitLabの「Premium License」を活用することで開発者のみアクセス可能な権限設定が可能となり、さらに完成したアセットは、コミッターがチェックしてからアセット公開用のGitLabへPushする運用体制となっている。
一方、アセット公開用GitLabはAWS上に構築されている。こちらも各地域のOktaと認証連携しており、「Ultimate License」を活用して、全社員がアクセス可能な状態となっている。
プラットフォーム構築で工夫した点が大きく3つある。1点目は、「将来的な拡張性を考慮に入れて構築を行った」点だ。プラットフォームはミニマルスタートとしたいが、将来的に重要インフラになるため拡張を考慮し、冗長構成への移行をスムーズにしたいと考えた。そこで、複数のコンテナを統合管理できるKubernetesを利用し、将来的な拡張を見据えた冗長構成への移行に対応できるように工夫した。
2点目は公開用GitLabについて「メンテナンス性を考慮に入れた」点だ。公開後に「ドキュメントがなくなってしまった」とか、「開発者がいなくなってしまってメンテナンスができない」といったことがないように、Infrastructure as Code（IaC）を全面的に採用してコードベースで管理することで、開発したソフトウェアがレガシーシステムとならないよう構築を自動化する仕組みを導入した。
3点目は公開用GitLabにおける「ゼロトラスト思想」だ。ID管理をIDaaS（Identity as a Service）によるクラウドベースの認証連携を利用前提とした構成とし、さらにサイバー攻撃者からのネットワーク到達性を考慮し、WAF（Web Application Firewall）を採用してセキュリティ性を高めた。
続いて、構築で苦労した点については、「既存GitLab環境とOktaの認証連携」が挙げられる。プラットフォーム構築に際し、アセット開発用の既存のGitLab環境にOktaを接続したときに、既存の認証に影響が及ぶことが懸念された。既存のGitLab環境はすでに開発で利用しているため、影響を最小限にする必要もあった。
そこで、検証環境として簡易な認証環境を準備し検証を行ったところ、メールアドレスの重複がなければ、既存の認証に問題がないことが確認され事なきを得た。

今後の展望・課題

今後の展望や課題については大きく3つのポイントがある。
1つは「公開用リポジトリーと開発用リポジトリーの集約・分散の議論」だ。現状は、公開用リポジトリーと開発用リポジトリーが別々になっているが、将来を見据えてこのままの運用でよいのかという議論である。
GitLabの「SSoT（Single Source of Truth）」の考え方に基づくのであればプラットフォームは集約することが望ましい。一方で、将来を見据えると、エンジニア以外の利用者がアクセスする機会も増えていく。アセット流通の観点からは、本当に必要で有益情報だけを参照できる現状の分散運用の方が望ましいという意見もあり、「集約」か「分散」の話については引き続き議論のテーマとなると考えられる。
2つめは「PR活動とビジネス成果の創出」だ。今後は社内へのPR活動を積極的に行い、アセット流通（開発・共有）の促進とともに、各地域が持っているアセットの活用（集約、登録）の促進を図っていく必要がある。
そして、同時にどれだけアセットが活用されたかトレースする仕組みを導入し、アセット流通がどの程度ビジネス貢献に寄与したか、可視化し正しく評価できる仕組みも必要だ。
3つめは「実行環境との連動」だ。ソースコードのリポジトリーなどの「アセットの共有」だけでなく、「動くモノ」としての共有を要望する意見が多く寄せられている。そこで、GitLabを起点とした実行環境をデプロイし、「どうやって動くのか」「どうやって活用するのか」といったところまで共有できるよう、拡張していきたいと考えている。

NTTデータについて

NTTデータは世界各国でビジネスを展開しており、グループ全体で18万人超の従業員を擁する。そして、デジタル領域における更なる競争力をめざし、中期経営計画における戦略の1つとして「アセットベースのビジネスモデルへの進化」を掲げている。

株式会社NTTデータ
技術革新統括本部
企画部
デジタルワークプレイス推進室
課長代理
新井 雄介 氏

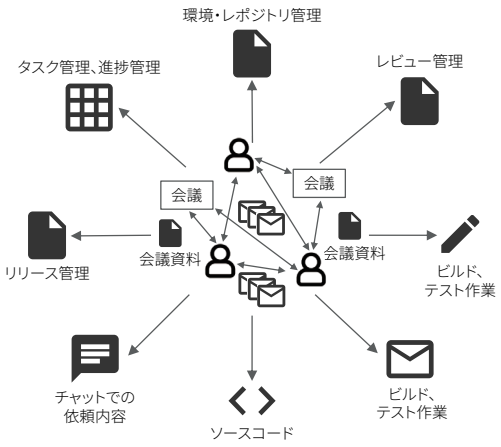
2008年4月 株式会社NTTデータ 入社
社内情報システム部門において、アプリケーション開発プロジェクトを中心に参画。また、商用プロジェクトにおけるCI/CD導入支援にも従事。
2018年以降は、海外を含めたNTTデータグループ共通システムの企画検討やグローバル開発プロジェクトにおけるプロジェクトマネジメントに従事。

オーケー株式会社が語るDevOps
GitLabの導入がもたらした品質と生産性向上とは？

高品質・Everyday Low Price

GitLab導入前の問題

当社は、ECなど事業の拡大、店舗数や販売エリアの拡大に耐えられるシステム構築を推進しており、その一環として開発体制の見直しが課題となっている。
店舗については、現状、関東圏内、主に神奈川県を中心に店舗を展開しているが、他の地方都市へ展開し、リージョナルチェーンからナショナルチェーンへ規模の拡大を目ざしている。また、コロナ禍も相まって、ネットスーパーの需要は拡大傾向にあり、ネットスーパー専用商品を発注するためのシステム刷新が必要だ。こうした背景から各種システムの強化が急務となっており、従来のオンプレミス環境からクラウド環境へシフト・刷新を行い、併せて開発環境もクラウドネイティブ化することが求められている。
一方で、ソフトウェア開発における課題は、タスク管理、ソースコード、ドキュメントなどの環境・リポジトリ管理、レビュー管理、リリース管理に別々のツールを利用しており、さまざまな問題が顕在化していた。
タスク管理はRedmineで行っていたが、ソースコード、ドキュメントとのひも付きがないため、管理が複雑であった。案件チケットは膨大にあるが、完了していても実際はレビュー中になっていたり、未完了のままになっていたりするなど進捗に信憑性がなかった。
また、環境・リポジトリ管理はSVNで行っていたものの、「どこに何があるのかわかりにくい」「履歴を追えばわかるが、履歴を追いかける作業に工数を奪われる」などの問題があった。
ソースコードなどのレビュー管理は独自の開発支援ツールで管理していた。DBに各種情報（ドキュメント、ソースコード、レビュー記録など）が蓄積されるものの、「レビューの何％が終わっているのか、状況が可視化できない」「SVNにコミットしないと反映されないため、履歴が煩雑になる」などの問題があった。
そして、リリース管理については、SVN、Fraqta、Redmine、Jenkinsなど複数のツールを組み合わせて管理していた。本番環境へリリースするためのディレクトリに格納（SVN）、格納された情報からリリースノートを作成し（Fraqta）、Redmineにてリリース用チケットを作成し（Redmine）、JenkinsがRedmineのチケット内容（リリースノート）に従って本番環境へデプロイするという流れだ。そのため、リリース資材管理が複雑となり、資材のチェックなどの工数が肥大化した結果、ミスを誘発し品質の低下を招いていた。



<u>情報源の分散化によるコミュニケーションコストの増加</u>
メール、チャット、チケットなど複数のコミュニケーションツールを通じて複数回やり取りを行う

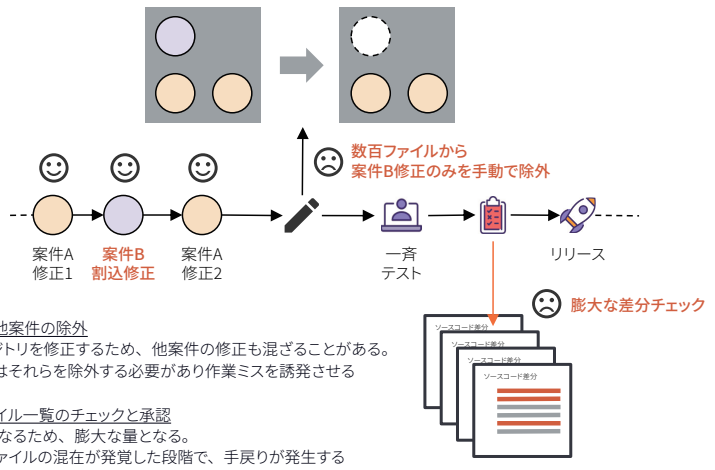
<u>進捗の手動更新</u>
実際の開発作業がチケットと連携していないため、作業の進捗や作業量などが手動更新になり、進捗管理の品質を低下させてしまう

課題①「管理工数の肥大化」と「管理品質の低下」

このように、人手で情報精度を維持することを前提としているため、信頼度の低い情報が散在し、情報確認のための追加作業コストが発生していた。その結果、次の2つの課題があった。1つめは「管理工数の肥大化と管理品質の低下」である。情報の信頼性を高める為に、管理者が現場エンジニア全員に対し、個別にヒアリングして、進捗管理を行った結果、莫大な時間を要することとなり、工数増大と管理品質の低下を招いていた。

課題②テスト前、リリース前の膨大な手動作業

2つめは「テスト前、リリース前の膨大な手動作業」の発生だ。複数案件で同一リポジトリを修正するため、たとえば、Aの修正を行うときにBのモジュールを除外せざるを得ない状況だった。他案件の修正も混ざることがあり、テスト、リリース時には手動かつ属人的な作業が必要となり、作業ミスを誘発させる結果となった。
また、リリース前の修正ファイル一覧のチェックが膨大な量となり、リリース対象外のファイルが混在した場合、リリースの作成とテストの手戻りが発生するなどの状況が発生していたのだ。



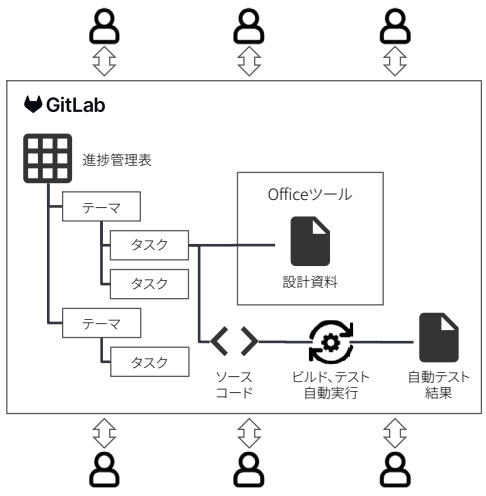
<u>テスト、リリース前の他案件の除外</u>
複数案件で同一リポジトリを修正するため、他案件の修正も混ざることがある。
テスト、リリース時にはそれらを除外する必要があり作業ミスを誘発させる

<u>リリース前の修正ファイル一覧のチェックと承認</u>
リリース直前の確認となるため、膨大な量となる。
またリリース対象外ファイルの混在が発覚した段階で、手戻りが発生する

課題の解決策とその効果

こうした課題を解決するため、当社はGitLabを導入し、タスク管理、リポジトリ管理、環境管理、リリース管理など、全ての開発プロセスやそれに伴う情報をGitLabに集約、一元管理することで、ミスと工数を削減したいと考えた。

GitLab導入により想定される効果は次の2点だ。1つめは「コミュニケーションコストの最小化と品質の向上」。情報が集約され、正しい状況が即時に把握できるため、コミュニケーションコストの最小化が図れる。また、作業結果が自動的に反映されるため、信頼性が高く効率の高い進捗管理が可能になる。
2つめは「リリース効率化による作業品質の向上」。当社は、GitLab Flowを活用しており、これにより、小さい単位、案件ごとに修正管理、自動検証、修正レビューと承認を効率かつ自動的に実施できることが期待できる。
具体的に、GitLabを導入してよかったポイントは次の5点だ。
1つめは「Issueと成果物が紐づいている」点だ。Issueに記載の作業、時間に対して成果物（具体的な差分）が完全に紐づいており、「タスクに対して適切な成果物ができたか」「時間に対して相応な作業量があったか」ということが一画面で確認可能となった。



2つめは「障害発生点がすぐ特定できる」点だ。開発者が、コミットしてすぐにCI/CDが回るため、CI/CDでエラーが発生した際、障害の起きたブランチ、それに紐づくIssueがすぐに発見できるようになった。
3つめは「個々の作業が明確となる」点だ。Issueボードによって、作業者はイシューをベースに作業できるようになり、リーダー以上の管理者はマージリクエストをベースに仕事ができるようになった。これにより、実施中の作業やこれからやる作業について、実態に沿った状況可視化が可能になった。
4つめは「どこに時間がかかっているかをGitLabが自動で集計してくれる」点だ。この機能を使えば、計画段階で見積もった時間と、実際にかかった時間の乖離を把握でき、次の作業予定にそれを反映することができる。
そして、5つめが「個人のアクティビティが確認できる」点だ。個人に着目して活動量を可視化し、最近取り組んだ作業の内容・成果物を一覧で確認できる。これにより、リモートワークでの信頼性向上を図りたいと考えている。

GitLabを利用してみて感じたこと

GitLabを実際に使用して感じたことは、「Issueが非常に使いやすい」という点だ。Issueがソースコード、ドキュメントなどの成果物と紐づいているため、進捗管理表など別で起こす必要がない。また、それらをタスクとして、カンバンボード上でメンバーに割り振ることができるので、タスクの状況も見やすく、進捗の遅れがすぐに分かる。
また、環境、リポジトリ管理については、「GUIにすぐれ開発環境への習熟に時間を要さない」点が便利だった。権限に応じた操作を付与でき、誤って本番環境へデプロイすることがないため、外部ベンダーに構築を依頼しやすくなった。また、ドキュメントがソースコードの近く（GitLab内）に構築でき、開発標準などを開発者がいつでも見られるのは大きなポイントだ。
レビュー管理については、レビュー依頼がマージリクエストの通知で気づくことができるようになった。また、Slackやメールでも通知が可能で、レビュアーはどのコードをレビューすればよいかが分かるため、無駄なやり取りが不要になった。

まとめ

このように、当社では、「管理工数の肥大化と管理品質の低下」「テスト前、リリース前の膨大な手動作業」という課題に対し、全ての情報をGitLabに集約し、開発環境とCI/CDをGitLabで一元管理することに取り組んだ。
これにより、「無駄のないコミュニケーションの実現」「管理情報の精度を維持し信頼度の高い情報を確認」「複数案件の柔軟なシステム開発を実現」といったメリットが得られた。
実際に導入して感じたことは、「無駄なコミュニケーションがなくなった」という実感だ。GitLabはまだ使い始めたばかりであり、今後はCI/CD、開発モデルは何が最適かなどの探求を続け、さらなる開発の標準化を進めていきたいと考えている。

オーケー株式会社について

オーケー株式会社は「高品質・Everyday Low Price」を経営方針として掲げ、1都3県を中心に、約138店舗のディスカウント・スーパーマーケットを展開している。また、2022年度JCSI顧客満足度調査において、オーケーがスーパーマーケット業種で12年連続1位を獲得している。

オーケー株式会社
IT本部業務システム部
マネージャー
金子 真三 氏

2017年10月オーケー株式会社入社。
入社後、自社システムの自動発注システムの運用保守リーダーとして従事。
2019年に自社物流センター設立、稼働に伴い、物流センターの管理システムの新規構築と物流センター向けに自動発注システム刷新プロジェクトのプロジェクトリーダーとして従事。2020年以降、上記システムの運用管理と、CI/CDのCCoE担当として従事。
現在、社内システムのクラウド化を推進中。

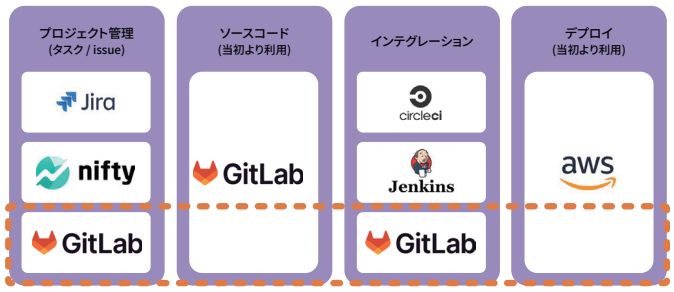
DevSecOpsで実現する生産性と品質の向上
カーフロンティアがめざす『開発のモダン化』とは？

CARFRONTIER

開発のモダン化をめざす我々がGitLabを採用した背景

秋山氏：我々は、自動車関連のデジタルサービスを約5年続けてきた。この間、主に各サービスの機能開発に注力してきたが、インフラや開発プロセスなどを改善することで、さらなる生産性や品質の向上が期待でき、競争優位性を高めることができると考えた。
というのも、機能開発に照準を合わせるあまり、開発プロセスにおいて手作業が多くミスが発生することや、リリース作業が非常に複雑になるケースがあるなどの弊害が出ていたからだ。また、インフラ面においても、CM実施後にアクセスが急増し、処理しきれなくなるなどの課題を抱えていた。これらの問題を解消するために、我々は開発プロセスの革新、モダン化をめざすこととなった。
開発のモダン化を進めるための指針として、我々は、Cloud Native Computing Foundationが提唱する「Cloud Native Trail Map」をベンチマークとした。一方で、Cloud Native Trail Mapは技術的なステップ論であるため、それを実現するためのプロセス改善も必要だと考えた。開発のモダン化には、技術面だけでなく管理プロセスの効率化を両輪で進めていくことが必要だからだ。
我々はWebサービス会社で、Webディレクターやエンジニアなど様々な職種の社員がいる。様々な職種の人が関わる開発プロセスをトータルに管理することが、品質の担保や生産性の向上につながると考える。
続いて、有料版のGitLab（Ultimate SaaS版）を選定した理由についてだ。当初、当社ではGitLab（SaaS版）をソースコード管理のためのリポジトリとして利用しており、各種Webサービスを提供するインフラはAWSを利用していた。この前提で開発プロセスの改善を行うに際し、プロジェクト管理ツール、CI/CDツールは別のツールを検討していた。
しかし、GitLab社との対話や、トライアルなどを通じて理解が深まり、有料版のUltimate SaaS版を採用することでプロジェクト管理もCI/CD も担うことができると考え、導入を決めた。

開発プロセスの改善（コンテナ化・CI/CDやプロジェクト管理の効率化）を図るに当たり、
ツールの検討も実施。最終的に、Gitlabで統一することを選択。



プロジェクト管理における我々の取り組み

続いて、プロジェクト管理におけるGitLab導入の効果について話す。当社のサービス企画から実装の基本的な流れは、主にWebディレクターが企画・要件定義したものをエンジニアで設計・開発・実装していく。
GitLab導入前の状況は、様々な情報が、様々なサービス・場所に、様々な形で管理されていた。プロジェクト管理者は企画の始まりからプロダクトのリリースまでを管理する必要があるが、そのためにいくつものツールの情報を参照して、コミュニケーションを行う必要があった。
また、プロジェクトに途中で参加した者は、これまでの経緯と決定事項の把握や自分の担当外の関連資料がどこにあるのかを把握することに非常に苦労していた。これにより、次のようなプロジェクト管理上の課題があった。

- 職種ごと、サービスごとに管理手法・粒度が異なるため、プロジェクト全体を把握するのが困難
- 企画がリリースされるまでの流れを一つの系譜で確認できず、リリースに至るまでに発生した資料を後の工程から探し出せない
- 過去の経緯は開発担当者の記憶に頼らざるを得ない
- 1つのプロジェクトを管理確認するのに4つのサービス、さらに担当者お気に入りのツールなどが乱立
- 非エンジニアメンバーからはRedmineのUIが不人気

GitLab導入後は、特にラベルの運用ルールを詳細に決め、次のような施策に取り組んだ。

- プロダクトに関わる事は機能改修の有無に関係なくissueを起票
- 企画時の目的と結論は必ずissueに記載（テンプレート化）
- 必要に応じてissueはEpicで束ねる
- 検討やQAはなるべくコメントして残す
- マージリクエストは、issueと関連付ける
- 役割に応じて、見たい粒度が異なる場合は、ラベルを活用し個人に最適なボードを作成する

その結果、3つのメリットが得られた。1つめは、「企画の始まりからソースコード、デプロイまで関連性が明確化した」ことだ。Issueには、企画の目的や背景、変更の概要などが記載されており、Issue内のリンクをたどれば、ソースコードの変更箇所や、デプロイ作業の進捗なども容易に確認することができる。これにより、企画の始まりからソースコードのリリースまでの経緯をGitLab上で可視化できるようになった。
2つめは「issueボードの運用によるタスク・issueの可視化」だ。我々はグループラベルとして10種類、計66個のラベルを作成している。同じissueでもディレクターが見たい切り口、エンジニアが見たい切り口、関連サービス担当者が見たい切り口は異なる。そこで、事前に決めたルールに則って各issueにラベルを貼り、タスクの種類、関連サービス、担当者、進捗状況等が見えるようにした。
3つめは「重要な情報はGitLabから確認できるというマインドの醸成」だ。ラベルを適切に配置することで、進捗ステータスはどこか、優先度はどうかをissueボードで可視化できるようにした。個人は必要に応じてボードを作ることもできるし、ラベル条件で全てのプロジェクトからissueを抽出することも可能になった。
これらをプロジェクト管理に役立てている。

開発プロセスにおける我々の取り組み

木村氏：続いて、開発プロセス改善の取り組みについて紹介する。
当社はこれまで、GitLabはバージョン管理を行うのがメインであり、単体テストはほぼ行っておらず、Amazon EC2へは手動でデプロイするという流れがメインだった。1年前より、GitLabのCI/CD機能を活用し、バージョン管理は従来通り実施しつつ、テスト、Dockerイメージ作成、Amazon ECRにプッシュするまでをGitLabのCIパイプライン上で行い、さらにAWS CodeDeployを用いてAmazon ECSに展開するという流れに改善を進めている。
また、Infrastructure as Code（IaC）にも取り組み、GitLab上でコードを構成管理からデプロイまで実施できる状態をめざしている。現在、当社のプロダクトの半分が新たなCI/CDプロセスに移行済みであり、残りの半分は年度内のリリースに向けて目下作業中という状況だ。
GitLab本格活用前の開発プロセスの課題は、「生産性と品質の改善をしなければ、ビジネスが求めるスピードに対応できない」というものだった。
具体的な課題は3つあり、1つめは「開発者が独自のテストを行っており品質が開発者に委ねられていた」課題だ。このため、テストの再現性が担保されておらず、毎回同じことを繰り返す必要があった。また、テスト実施者によってテストの品質もばらつきがあり、自動化も実施していないため、テストの工数が非常にかかる状況だった。
2つめは「ブランチ運用がルール化されていない」課題だ。組織内、開発者それぞれで「ブランチの切り方やコメントの書き方などが異なる」状況で、経緯が追えない、意図がわからないといったことも頻繁に発生していた。
3つめは「リリースフローが属人的」という課題だ。毎回、手動でリリースを行っており、人為的ミスが発生しやすく、自動化されていないためリリースのたびに関係者の工数がかかる状況だった。
これらの課題に対して、我々がGitLabを使ってどのような取り組みを行ったかを説明する。
1つめは「コンテナ化とテスト自動化、CI/CDパイプラインの構築」だ。まず、各サービスを担うアプリケーション群をコンテナ化することを考えた。これに伴い、本番実行環境としてAmazon ECS / Fargateを採用、インフラ環境もIaCで構築することにチャレンジしている。
並行して、開発チームでは書いていなかった単体テストも少しずつ書き始めた。さらに、これらのテストやビルド、プッシュ、デプロイまでを自動化するためにGitLabパイプラインにて接続することに取り組んでいる。
テストという観点では、自動テストツールの「Autify」を利用してEnd to Endテストを自動化することにも取り組み、GitLabパイプラインでの連携にもチャレンジしている。また、可観測性を実現するため、ログ一元管理にSaaS型サーバ監視サービス「NewRelic」を導入した。
2つめの取り組みは「ブランチ運用の改善」だ。GitLabフローを導入したり、ブランチの命名規則とコミット時のコメントの運用ルール化を進めたりしている。
これらの取り組みにより、GitLabにはCI/CDや、Docker、DevSecOpsなどの考え方が導入されており、使っているうちに自然とモダンな開発プロセスに変わらざるを得ないという学びを得ることができた。
そして、得られたメリットは「作業品質を含む、全体的な品質向上」と「属人化の改善、効率化が進んだこと」が挙げられる。定量的には、概算ではあるものの、ユニットテストのカバレッジは、当社全体で0だったものが約10%に進捗し、一部のAPIサブシステムについては80%に到達したものもある。またデプロイまでの手順も、手順数にして約80%削減できた。
なお、インフラの維持コストについては、夜間の検証環境電源管理やコンテナ化、AWS Auto Scalingの実装により、最適な規模のAWSリソースで運用できる状態となり、コスト削減にもつながっている。
開発プロセス改善はまだ道半ばではあるものの、ここまでの成功要因を総括すると以下のポイントが挙げられる。
1つめは、チャレンジを理解し予算を捻出してくれた経営層の協力。2つめは、ともに考え、道を切り開き、知識とスキルをチームに展開してくれるエンジニアの存在。そして3つめは、開発プロセスをモダン化したい、変わりたい、変えたいと覚悟を持って取り組んだクリエイティブチームメンバーである。今回は、改めて全社的なチャレンジであったと感じている。

GitLabを使って解決したい今後の課題

最後に、「GitLabを使ってこんなことをしていきたい」という展望を説明する。さらなる生産性と品質の向上をめざし、ビジネスにおいてもコストセンターではなく利益を生み出すプロフィットセンターとなれるよう、チームとして変革していきたいと考えている。
そのプロセスとして、まだまだ「運用ルールの統一」が必要だ。限られたリソースで、やることの選択と集中を実現できるチームにするためにも、担当者間の業務知識の平準化を進め、メンバーのローテーションや、横断して各サービスを見られるようにするために、引き続き推進していきたい課題だ。
そして、「DevSecOpsのさらなる推進」も必要だ。生産性の向上はいうまでもないが、Shift-Leftによって得られる効果や、トータルでのコスト削減効果は大きい。今後はセキュリティの領域を早急に組み込むべくGitLabの機能をさらに活用していきたいと考えている。
この1年、GitLabを使うことでDevOpsへの取り組みが大きく進んだ。今後は、セキュリティの強化に取り組み、DevSecOpsを実現し、さらなる生産性と品質の向上をめざしていきたいと考えている。

株式会社カーフロンティアについて

カーフロンティアは1988年設立、自動車整備・メンテナンスに関するWebサービスを提供する三菱商事グループの会社だ。昨今、様々な分野でデジタル化が進んでいるが、自動車のメンテナンスに関しては、まだまだアナログな部分が多く、そこをデジタル化することで新たな価値を提供しようとしている。運営サービスについては、たとえば、「timy（タイミー）」がある。これは、エンドユーザー向けに車検やタイヤ交換などのメンテナンス予約ができるサービスだ。我々は、ガソリンスタンドなどのカーメンテナンス事業を担う加盟店舗を保持し、オンラインからの送客や、実店舗支援を担う各種サービスを展開している。

株式会社カーフロンティア
Application Development Unit
システムエンジニア
秋山 達郎 氏

大学卒業後、自動車関連の組み込み開発からキャリアをスタートさせる。基本設計から実装、テスト、保守及び顧客からの問い合わせ対応などを通じて、IT開発プロセス全般に関する知見を習得する。 その後、Androidを活用した開発において、開発環境構築、インフラ構築に従事することでアプリケーションからインフラに関して、活動の幅を広げた。その後、AWSを活用した開発案件に従事し、処理時間改善などのアーキテクチャ選定にも関わる。 カーフロンティア入社後は、新技術領域における開発を主担当者、DWH開発担当として活躍している。アプリケーション開発における、業務フローの確立にも従事することで、Gitlabを活用したエンジニア組織の生産性向上にも活躍の幅を広げている。

株式会社カーフロンティア
Site Reliability Engineering Unit
システムエンジニア
木村 純也 氏

和食の料理人からという異色のキャリアの持ち主。IT業界に対する将来性を感じ、SESからIT業界におけるキャリアをスタートさせた。システムの監視、運用、保守を通じて、インフラ業務における知見を得る。その後、サーバ設計、ネットワーク設計及び運用・保守を担うとともに、新人向けの講師を歴任。
その後、Azureを活用したインフラ設計、構築、運用業務とパブリッククラウドを活用する知見を得る。IaCを用いたインフラ構築の自動化なども 担うことで、生産性向上にも寄与。その後は、ビジネス理解を深めるため、Webサービスのディレクタ兼インフラエンジニアを担う。 カーフロンティア入社後は、AWSの費用削減案立案・実行、監視行動化、インフラ運用の自動化など、生産性向上と費用削減の両立、セキュリティ 向上などの課題解決に挑んでいる。

9年間使ってわかったGitLabとDevOpsの話

ZYYX

ジークスがGitLabを選んだ2つの理由

当社は2013年から9年間、GitLabを活用している。当初はソースコードの管理がメインだったものの、現在ではCI/CDやセキュリティなど、DevOpsにまつわる全てのプロセスに至るまで活用の幅を広げている。
当社がGitLabを選んだ理由は大きく次の2つある。
1つは「高いセキュリティ性」だ。取引先企業は多種多様な業種・規模にわたるため、ジークスにとって最優先事項はクライアントのソースコードおよびデジタル資産を高い安全性で保護することにある。
2つめは「CI/CD」だ。限られた開発メンバーでより高いパフォーマンスを発揮するためには、DevOps、特にCI/CDによる時間とコストの節約が不可欠だった。

高いセキュリティ性

では、それぞれのポイントについて詳しく話していく。「高いセキュリティ性」について、最初にGitHubとGitLabの誕生背景の違いをおさらいすると、GitHubは、2008年にリリースされ、リリース直後からOSSの開発で使われはじめた。こうした背景から、パブリックリポジトリが活発的だという特徴がある。
そして、GitLabは2011年にプライベートリポジトリのためのOSSとしてリリースされた。そのため、自社管理サーバーにインストールして独自運用ができることが特徴だ。
ジークスでは2013年11月にGitLabを採用した。当初はオンプレミスによる独自運用で、2016年7月には月1回のバージョン更新サイクルになり、2018年9月にはメンテナンス作業の自動化を実現するなど拡張を続けてきた。
しかし、運用を続ける中でサーバー老朽化とそれに伴う保守コストの高騰が課題となり、2020年にはプラットフォームの移行が検討された。検討段階では、GitHubや、AWS上にGitLabをインストールする運用方法も検討されたが、最終的にはSaaS版のGitLabに移行して現在に至っている。
SaaS版のGitLabに移行した決め手となったポイントは次のとおりだ。

- 独自運用と同等のセキュリティを担保できる
- （SaaS版による）メンテナンスフリーと優先度の高いプライオリティサポートが受けられる
- 足りない要件はAPI連携でカバーできる

そして、自社運用からSaaS版への移行時に気をつけたことは次のとおりだ。これまでの独自運用では全データがオンプレミスサーバー上にあり、当社が管理者として管理していた。しかし、SaaS版への移行によりデータはGitLabが管轄するクラウド上で保管されるため、データ保護に関する各自のリテラシー意識がより大事になる。
そこで、パブリックリポジトリによる情報漏えいが起きないよう運用ルールを徹底させるとともに、社内ポータルでの周知や、ISMS（情報セキュリティマネジメントシステム）教育を充実させるなどの施策を行った。
また、運用ルールが個人依存になるのを防ぐための対応として、Slack BotとGitLab API連携によるセキュリティ監視の強化も行った。たとえば、パブリックリポジトリがメンバーで作成されていないかのチェックを行うことや、運用ルール外の検知があったときにSlack宛に通知が飛ぶよう設定、あるいは、アカウントの参加・退出ステータスの通知やシステムインシデント情報の通知もSlack宛に設定した。

これらの施策により、高いセキュリティを保ちつつ、メンテナンスフリーで運用することが可能になった。

GitLab CI/CD

続いて、CI/CDの観点からSaaS版GitLabの移行ポイントを解説する。
DevOpsやCI/CDは、ソフトウェア開発の効率化のための手法だ。ソフトウェア開発でよくある課題として、たとえばリリース時には、「リリース作業が特定の人に依存している」「引き継ぎたいが時間が取れない」「手順書はあるが人為ミスが発生する」「久しぶりのリリース作業で心理的に不安」といった課題が挙げられる。
また、テスト時には、「リリース後に同じテストを繰り返している」「テスト不足でバグ検知が遅れる」などの課題が挙げられる。
CI/CDは、こうした課題を解決するために、作業を自動化して多くの時間節約と人為ミス軽減を実現するものだ。そして、GitLabを活用したCI/CDなら、ツールの切り替えが不要で、さらにSaaS版はCI/CD実行サーバーの準備も不要といったメリットにより、誰でもすぐに使い始められるというのが選定の大きな決め手となった。

ジークスにおけるCI/CD導入成果

現在、ジークス全体の総CI/CD実行時間は月間5,000分にのぼる。たとえば、2022年1月から9月の実績では、CI/CD実行時間は毎月平均で5,000分、最も多い月で月間7,850分であった。これらのCI/CDによる時間節約効果は、1日8時間で換算すると平均で月10日、最大16日分に相当するものだ。
また、あるプロジェクトにおける1ヵ月のCI/CD実行時間は20時間となっている。GitLab導入により、ビルド・デプロイの自動化による、安定かつ迅速なリリース作業が可能になったことや、リリース後の動作テストも自動化することで、バグの早期発見が可能になった効果が得られた。
一方で、ジークスの全プロジェクトに対するCI/CD稼働率は、わずか4.5%に過ぎない。
4.5%の稼働率で月間平均10営業日相当の時間が節約できていることを考えれば、プロジェクトにおけるCI/CD稼働率を伸ばすことで、開発効率もビジネスメリットもさらに高まることが考えられる。
そこで当社では、将来的な目標として現在の6倍の「稼働率27%」という目標を掲げている。これにより、月間稼働日を20日とした場合の節約効果である「月間10日＝0.5人分」の節約効果が、「月間60日＝3人分」の節約効果まで引き上げられることを見込んでいる。
最後に、CI/CD稼働率を引き上げるための取り組みを紹介する。まず、2022年6月には「SaaS GitLab（Premium）」にアップグレードを行った。
また、GitLab日本法人によるサポートにより、社内向けのワークショップを開催した。これまでは英語のドキュメントを各自で調べて導入設定などを行っていたものが、GitLabによるサポートにより、日本語ドキュメントでCI/CDに関する知見を深めることができた。
そして、ジークス内での活動としては、ナレッジ共有やプロジェクトごとの個別導入サポート、そして定期的な勉強会や小委員会によるプロジェクトごとのサポート分担、フォローアップなどの社内啓発のための仕組みづくりを行っているところである。

ジークスについて

ジークスは、「デザイン力× 技術力のパフォーマーとしてITを社会に実装する」をミッションに掲げる、80名規模のソフトウェア開発会社だ。社員の約半数はサーバーサイドエンジニアやアプリ開発エンジニアであり、実績はUX/UIデザインから大規模アプリ・システム開発まで多岐にわたる。

ジークスとGitLabの歴史

■オンプレミスでの独自運用
2013年11月v6.2.4 初回インストール
2016年7月v8.6.4 月一回のバージョン更新サイクルに
2018年9月v11.2.3 メンテナンス作業の自動化

■サーバー老朽化と保守コスト高による移行検討
2020年6月v13.0.7SaaS GitLab (Bronze) に移行
2022年6月v15.1.0SaaS GitLab (Premium) にアップグレード

株式会社ジークス
新規事業開発室
株式会社ウーブ
執行役員
プロダクト責任者
久保 仁詩 氏

エンジニアとしてフロントエンド・サーバーサイド・インフラまで幅広く経験。
現在はその経験を活かし、新規事業の立ち上げ・プロダクト開発に従事。ジークスのGitLab運用・保守を担当。

Gitを前提とした開発標準ルールの策定と開発効率化
GitLabによるDevOpsを見据えた開発フローの標準化

TOPPAN

これまでの開発における課題

当部門では、行政・金融機関向け提供サービスのシステム開発／運用保守を担っており、100を超えるシステムと顧客が存在しています。これまでの開発では、プロジェクトごとに、開発ルールや各システムの資源の管理ルール、ツールが存在しておりました。そのような状況の中で、近年より求められる、高品質で迅速なシステム開発体制を実現するためには、いくつかの開発課題が存在していました。
具体的には次の3つです。

課題1. 案件個別にGitの運用ルールを策定・運用しており、知見の共有が困難
運用ルールの策定が案件ごとに必要なため、案件毎にルールの検討・導入工数が必要でした。また、ルールが案件ごとにガラパゴス化しており、他案件に知見を共有しづらくなっていました。更に担当者のスキルに依存した属人的な運用になっていました。

課題2. レガシーな手法による低効率な開発
当部門ではウォーターフォール型の開発手法を取ることが多く、各開発工程毎に計画した環境への開発資材のデプロイ、テストを実施する必要があります。このとき、開発資材のデプロイは手動で実施することが前提となることが殆どでした。そのため、以下のような工数やリスクを考慮した開発となり、コストや品質に影響する課題がありました。

- デプロイの手動実行ミス防止のための工数（ダブルチェックによるデプロイ等）
- 手動を前提としたリグレッションテスト（回帰テスト）を含む各種テストの工数
- ドキュメントや手順にない属人的な作業が入り込むリスク

課題3. 活用するツールが部門で統一されておらず、運用、増員／配置転換が高コスト
別の案件のヘルプや配置転換等の際に、案件毎のルールやツールの差異に対する学習コストに課題がありました。異なるUIに対する理解、複数ツールからの通知確認、アップグレード時の互換性確認、ツールごとの権限管理と統制、ステージ間の分析などは多くの工数を要します。さらには各ツール間でも独自の連携が存在していたため、運用コスト、増員・配置転換時の学習コストが高くなりがちでした。

課題に対する2つの施策

そこで、当部門では、まずは資源管理をGitに統一した上で開発ルールの標準化を行い、DevOpsプラットフォームである「GitLab」を導入することで、これらの課題にアプローチしてきました。
具体的には「これまでの開発課題を解決し、より高品質で迅速なシステム開発体制を実現すること」を目標に掲げ、改善に向けた2つの施策を行いました。

施策1. 開発標準フローを策定し、ナレッジ共有のための仕組みを構築する

1つめの施策は「開発標準フローを策定し、ナレッジ共有のための仕組みを構築すること」です。課題1に対する施策として実施し、具体的には次の6つのステップで施策を進めました。

ステップ1. Gitを利用した開発フローを検討
次の観点のもと、代表的な開発フローを調査し案件担当者間でフローを検討しました。

- 環境に合わせたブランチ定義が可能で、環境ごとのソース管理が容易であること
- 部門で扱うシステムの2割は開発が並行しており、並行開発に適応可能なこと
- 運用起因の障害を防ぐため、ブランチ管理が複雑でないこと

その結果、バランスの良い「GitFlow」をベースに、部門のGit開発フローとして整理しました。

ステップ2. 案件全体の標準開発フローを定義
各開発現場にヒアリングを行い、各案件の現状の開発を整理しました。整理した内容を元に案件全体の標準的な開発フローを定義しました。

ステップ3. ステップ1、2の両フローを統合し標準ルールを定義
ステップ1で整理したGitFlowベースの開発フローと、ステップ2で定義した案件全体の標準的な開発フローを統合し、開発標準フローとして策定しました。

ステップ4. 標準フローの導入説明会を実施
部門全体に導入説明会を実施し、導入方針と策定した開発標準フローを周知いたしました。

ステップ5. ナレッジ蓄積／共有のための仕組みを構築
要望・課題・QAを吸い上げ、ナレッジとして標準ルールにフィードバックし、改訂した標準ルールを部門へ共有できる仕組みをGitLab上に構築しました。これは本開発標準フローに対するQA・要望・課題をSlackで受け付け、それらをGitLabのIssueでリスト化して管理、現場にあったルールへ適宜更新していくものです。これにより、起票ハードルを下げつつ、先行してGitLabを活用することで続く施策への下地を醸成する狙いもありました。

ステップ6. 導入状況をモニタリング
標準フローと照らし合わせて定着状況を定期的にモニタリングし、必要に応じて改善依頼と支援を実施しました。継続的に行い、定めた標準ルールの定着を推進しております。

施策2. DevOpsによる迅速で高品質かつモダンな部門共通の開発基盤を構築

続いて、2つめの施策は「DevOpsによる迅速で高品質かつモダンな部門共通の開発基盤を構築すること」です。課題2、3に対する施策として実施し、次の5ステップで進めていきました。

ステップ1. Git関連サービス比較検討
GitLabと他サービスを比較し、部門共通で導入する基盤を検討しました。「機能」「CI/CD」「料金」「環境構築」「利用組織数」の5項目で比較検討した結果、プロジェクト管理もDevOpsを前提とした環境構築もオールインワンでできることが決め手となり、GitLabを選定しました。

ステップ2. GitLabにおける検討
「SaaS版とオンプレミス版」「フリーライセンスと有償ライセンス」の比較検討を実施しました。本検討については、運用負荷の低さからSaaS版を、そして、複数人での開発、DevOpsを前提としたプロジェクト管理を考慮し有償版（Premium版）を選定することとなりました。

ステップ3. GitLabにおけるDevOpsの整理
どのタスク（Issue）でコーディングし、誰がレビュー・承認して、ソースがマージされたか、一連の開発プロセスに対して、トレーサビリティを保ちながら推進していけるようGitLab導入方針を整理しました。また、CI/CDについては、できる案件から任意でスモールスタートしました。

ステップ4. 開発標準フローへのGitLab／DevOpsの適用
DevOpsのベストプラクティスを適用し、ステップ3で整理した導入方針に従い施策1で策定した開発標準フローをGitLabを活用して改善していく取り組みです。ウォーターフォール開発におけるGitLabでのタスク管理方針に従い、Epic、Milestone、Issueによってブランチ作成からマージ作業を含む一連のタスクを管理。また、GitLabによる主要ブランチのマージ必須化と承認ルールにより、開発標準フローをシステム化しました。

ステップ5. 案件テスト適用と開発標準フローブラッシュアップ
施策1で構築したナレッジ共有の仕組みを活用し、GitLab上で次期バージョンの開発標準フローを整備していく取り組みを行うものです。

施策に対する成果

施策1に対する成果は、部門標準となる開発標準フローが策定されることで、新規案件におけるルール策定・技術調査コストが低減したことが挙げられます。また、標準フローに対し課題や要望が挙がったものの、GitLab上に構築した仕組みにより迅速な回答・フローの改訂が行えるようになり、知見の共有・蓄積が可能になりました。
今後は、導入状況のモニタリングを進めながら、引き続きQAや要望・課題に対応することで、より使いやすい標準フローへと改善を繰り返し、プロジェクトへの導入率100%をめざしていきます。
続いて施策2に対する成果は、GitLabによるワンアプリケーションにより、DevOpsをテスト案件で部分的に実践することができました。また、開発標準フロー運用にあたっては「マージ先の誤指定の防止」「セルフマージによる担当者任せの改修の防止」などを仕組みとして実現することができました。

今後の展望

今後は、GitLabを活用した開発標準フローの改訂が完了次第、正式に部門展開を予定しております。またCI/CDを任意から必須の取り組みとすることで、DevOpsを推進し、より迅速で高品質な環境を構築していきたいと考えます。
それぞれの施策に対する社内の反応ですが、施策1では、導入説明会後、開発担当者から様々な声が挙がりました。たとえば、「請負契約において開発発注先にもルールを徹底させ、それを凸版が管理するのは無理がある」という意見に対しては、完全請負契約で凸版の責任範囲は納品物の管理のみである場合、納品物のバージョン管理としてのみGitで管理するよう、ルールを改訂して対応しました。
こうした一つ一つの対応によって、導入後にアンケートを実施した結果、導入開始から約3ヵ月で35%の案件に浸透していることが分かりました。今後は、継続的に要望・課題を吸い上げ、開発標準フローをブラッシュアップし各案件への浸透を促進していきたいと考えています。
施策2については、GitLabテスト適用後に、利便性や有効性についてアンケートを実施したところ「コードレビューがしやすく変更の確認もしやすい」「テストによる不具合起票からバグFIXまで、トレーサビリティが確保された対応が可能」といった声があがりました。今後は、正式な部門展開に向けて、わかりやすいマニュアルの整備とさらなる丁寧な説明が必要と考えています。

最後に

当部門では、「これまでの開発課題を解決し、より高品質で迅速なシステム開発体制を実現する」目標に対し、2つの施策に取り組んできました。
今後はその先にあるDevSecOpsを見据え、GitLab搭載のセキュリティテストを検討し、さらなる開発の高速化と品質向上を推進していきます。具体的には、静的スキャン（SAST）やコンテナスキャンを利用し、通常案件終盤で行われる脆弱性試験の工程をシフトレフトすることで、手戻りリスクを軽減し、品質向上を実現したいと考えています。
GitLabを活用し、DevOpsを実践して「開発をより楽に、より楽しく」を推進していきたいと考えています。

凸版印刷について

凸版印刷は、1900年（明治33年）に設立され、当時の最先端技術である「エルヘート凸版法」を基礎に、証券印刷やパッケージ印刷等の分野でビジネスを拡大してきました。
印刷を通じて培った「印刷テクノロジー」をベースに「情報コミュニケーション事業分野」「生活・産業事業分野」「エレクトロニクス事業分野」の3分野にわたり幅広い事業活動を展開しており、近年では、印刷にとらわれず情報の価値を最大化し、円滑なコミュニケーションを時代に合わせて提供。特に、デジタル技術を活用しビジネスや生活に変革をもたらすDX領域に注力しています。浅野氏の所属するハイブリッドBPO企画本部では、デジタルとアナログの両オペレーションのハイブリッド型サービスを展開し、個人情報などセキュアな情報を取り扱う行政・金融機関向けBPOを提供しています。

凸版印刷株式会社
情報コミュニケーション事業本部
セキュア事業部
ハイブリッドBPOセンター
ハイブリッドBPO企画本部
ICT開発二部
主任
浅野 翔太 氏 ※

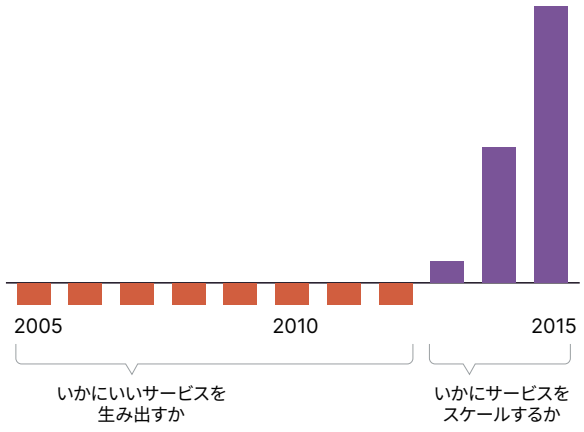
大手通信会社、金融機関向けアプリケーション開発に従事。Webアプリケーション開発を主軸にしつつ、スマートフォン向けアプリケーション開発も手掛ける。
近年は、行政・金融機関向けWebアプリケーション開発のための社内フレームワーク基盤構築プロジェクトのPM、および、社内におけるGITを前提とした開発業務フローの標準化・効率化プロジェクトのPMを兼任し、社内の開発現場の強化に邁進。

※ 2023年4月1日よりTOPPANエッジ株式会社 ハイブリッドBPO統括本部ICT開発本部所属

弁護士ドットコム

なぜ我々はGitLabを導入するに至ったのか

当社は、2005年の創業後、しばらく赤字が続き、2013年頃から軌道に乗って成長基調となった。赤字のときは「いかにいいサービスを生み出すか」が課題で、顧客課題を解決するサービスをいかに提供するかというPMF（プロダクトマーケットフィット）が大きなテーマだった。
しかし、成長期にあっては、いかにサービスをスケールするかが大きなテーマとなり、DevOpsの基盤として、2014年からGitLabを導入している。



導入前の課題には、いわゆる属人的な「カウボーイスタイル開発」が挙げられる。この開発スタイルは、いかによいサービスを生み出すかという時期は有効だったが、ビジネスをスケールさせるときにはいくつかの問題が生じた。
たとえば、「開発プロセスの不在」という課題だ。これまでは開発テーマが生じると、つどマスターに直接コミットして開発をプッシュするという開発スタイルで、「Redmine」「Review Board」といったツールがあったものの、利用は任意だった。
また「優先度付けの不在」という課題もあった。依頼開発がエンジニア個人に飛び交い、全てのタスクが最優先という状態で、また、jQuery Mobileなど技術的負債の解消も手つかずの状態だった。そして「貧弱な開発環境」という課題もある。テストサーバー1台を皆で共有していたため、誰かがデータベースのスキームを変更した時に、他のプログラムが動かなくなるなどの事象が起きていた。
これに対し、開発プロセスの整備として、開発プロセスの策定に着手。DevOpsプラットフォームとしてGitLabを導入し、MR（マージリクエスト）によるピアレビューを行うとともに、スクラム開発を導入した。また、全社での優先度付けを行い、スプリントごとに優先度を判断することにした。
そして、ローカル開発環境の整備も行い、「Vagrant」「Docker」「Ansible」の導入など、生産性を高める仕組みを整備した。

GitLab導入の決め手となったポイントについては、プラットフォームに求められる要件である「コード管理・レビューに関する機能が充実していること」「会社メールアドレスでアカウント管理できること」「（特にコスト面で）スモールスタートできること」の3点で最も優れていると判断したからだ。
そして、GitLab導入により「コードの一元化」が実現された。導入前は、Gitサーバーに一元化されてはいたものの、可視性が低かった。しかし、導入後は会社のコード資産が一元管理できるようになった。
また、「コードの権限管理」というポイントも挙げられる。導入前は全員がすべてのリポジトリにアクセスできたが、導入後は、リポジトリごとの権限管理がGUIから可能になった。
さらに、「コードレビューの必須化」というポイントもある。導入前は、レビューツールはあるものの、利用は任意だった。しかし、導入後は、すべての変更はマージリクエストを作ってレビューする文化が根づくこととなった。

GitLabの効果

コードの一元化	• 前: Gitサーバーに一元化されてはいたが、可視性が低い • 後: 会社のコード資産が一元管理できるように
コードの権限管理	• 前: 全員がすべてのリポジトリにアクセスできた • 後: リポジトリ毎の権限管理がGUIから可能に
コードレビューの必須化	• 前: レビューツールはあれど利用は任意 • 後: すべての変更はMerge Requestを作ってレビュー、という文化が根付く

我々はどのようにGitLabを活用しているか

現在のDevOpsの運用は、コードを作成し、AWS環境に反映。運用のデータは、クラウドサーバー監視ツールである「DataDog」で収集し、コードの修正はプロジェクト管理ツール「JIRA」でチケットを切って修正を行っている。
そして、コード管理でGitLabは大きな役割を果たしている。
コード管理を行うプロジェクト数は、導入前の2から500以上に増え、マージリクエストでのコードレビュー数は月間800件以上にのぼる。また、テスト数も大幅に増加しており、CI（継続的インテグレーション）におけるGitLab Runnerの数は月間で15000以上と、一つの変更に対して複数のパイプラインが走り、コードの健全性、安全性を確保している。
特に、GitLabに備わる、依存関係の更新チェックを自動化するツール「Renovate」により、変更に対して追従が可能になった。また、コード変更がそれほど大きくない場合、マージリクエストから開発者をランダムに選んでアサインができるようになったため、チームを超えたコラボレーションが可能になった。
このように、DevOpsにおけるGitLabの効果は「開発者体験の向上」「セキュリティの向上」にある。
前者としては、レビューやCIがGitLabに集約されることで素早く安全にコードを書けるようになり、Four Keys（ソフトウェア開発の生産性を測る4つの指標）における「デプロイ頻度」「変更のリードタイム」を高いレベルで維持できている。
後者としては、特にクラウドサインは顧客の契約書を預かるサービスなので、高いセキュリティレベルを求められるが、開発のアクティビティがGitLab上に集約し、それらをコントロールする機能が豊富である。また、Premiumプランでの機能となるが、監査ログ機能が充実しており、監査証跡としても活用することができる。
最近の開発プロセスにおいては、Code is SSoT（Code is Single Source of Truth）だといわれる。たとえば、パッケージの依存バージョンは、lock fileに記述される。コンテナもコードから冪等性を持って作成できるし、コンテナ基盤を活用する場合は一度作成したコンテナは変更しない「immutable workload」が自然だ。
一昔前であればサーバー上の設定は“秘伝のタレ”として受け継がれていたが、今はコードに表現される。さらに、「Policy as Code」として、プロセスを統制するポリシーまでもコードで表現するようになっている。
このようにコードがすべての起点となるため、その変更や管理を司るGitLabは、生産性の面からもセキュリティの面からも重要だということができる。

Code is SSoT	• package lock file, Immutable Workload, IaC, Policy as Code など • コードがすべての起点に
開発者体験	• レビューやCIにより安全に素早くコードが書ける • Four Keysの「デプロイ頻度」「変更のリードタイム」を高いレベルで維持できている
セキュリティ	• Merge Request の承認ルール(Premiumプラン) などワークフローの統制が可能 • 開発アクティビティが集約されたGitLabで、監査ログ(Premiumプラン) による証跡確保

我々はGitLabひいてはDevSecOpsとどのように向き合うのか

現代では、ビジネスの競争力を高める観点からもDevOpsは欠かせないものと認識されている。
企業は、DevOpsへ投資し、「変更のリードタイム」「デプロイ頻度」「MTTR：平均修復時間」「変更失敗率」というFour Keysの指標を高め、高速リリースサイクルを実現することが重要だ。
その点で、GitLabでコード管理がスムーズにできることは有効である。また、日本企業におけるDevOpsの導入はまだ10%程度といわれ、先んじた取り組みが差別化要因となる。
さらに、クラウド活用が進むにつれサイバーセキュリティの重要性が高まるのに伴い、セキュリティ対策もパラダイムシフトが求められている。従来のセキュリティ対策を、クラウドを前提として社内外のあらゆる領域が潜在的に危険であるという「ゼロトラスト」の考え方で再構築し、アプリケーションワークロードのセキュリティを高めることが必要となるだろう。
すなわち、開発プロセスの各段階で、セキュリティ対策を組み込んでいくDevSecOpsが重要となる。
クライアントサイド攻撃やサプライチェーン攻撃といった新たな脅威への対応も必要だ。前者は、CI/CD pipelineを狙った攻撃があるが、GitLabにはAWSのキーを持たずに、安全にAWSにデプロイできるようなOIDC連携機能があり、対応が可能だ。また、後者には依存ライブラリを狙った攻撃があるが、GitLabにはスキャン、モニタリングの機能を備えており、まだ定石となっていない新たな攻撃へ対応できるような機能を備えている。
現代ではビジネスの競争力を高めるために、DevSecOps への取り組みは欠かせない。しかし、DevSecOpsには唯一の正解があるのではなく、個社状況に応じた漸進的な取り組みになる。この点において、GitLabは豊富な機能でDevSecOps を支えてくれる。
DevSecOpsの実現方法に唯一の正解はなく、開発組織のフェーズやビジネスモデルなどにより異なる。自社に適したDevSecOpsツールへの投資が、その後のプロダクトやビジネスの方向性を左右すると言っても過言ではないだろう。

弁護士ドットコムについて

弁護士ドットコムは、2005年に創業し、「まだないやり方で、世界を前へ。」を掲げており、専門知とテクノロジーで社会に貢献するために、様々なソリューションやサービスを提供している。
無料法律相談ポータルサイト「弁護士ドットコム」や、企業法務ポータルサイト「BUSINESS LAWYERS」、また、税務相談ポータルサイト「税理士ドットコム」などのサービスを運営するほか、電子契約マネジメントプラットフォーム「クラウドサイン」というソリューションを提供している。

弁護士ドットコム株式会社
執行役員
技術戦略室長 兼
クラウドサイン事業本部副本部長
市橋 立 氏

2005年東京大学大学院工学系研究科卒業。
同年アクセンチュア株式会社入社、戦略グループ通信ハイテク事業本部コンサルタントとして新事業戦略・事業戦略・マーケティング戦略の立案および業務改革支援などに携わる。起業を経て、2014年1月に入社。
技術戦略室室長として全社のエンジニア・デザイナーを統括しつつ、2020年6月からクラウドサインのプロダクト開発の統括も兼務。

フィックスターズでのGitLab活用の道のり
データ過多時代の高速ソフトウェア開発チームの実践例

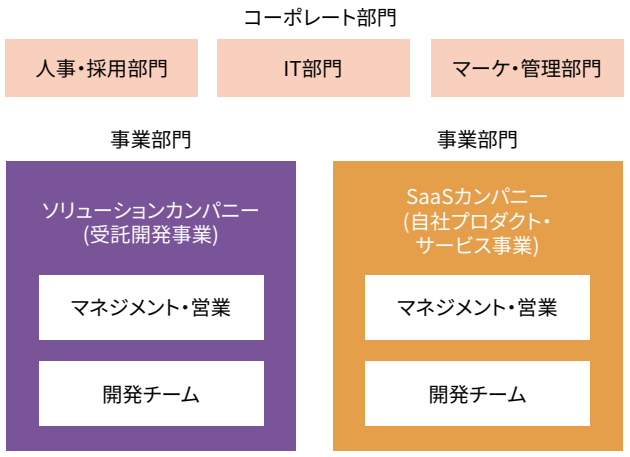
GitLabの導入

孫氏：フィックスターズの社内開発インフラは、2002年の設立後しばらくはTrac（チケット管理システム）やSubversion（バージョン管理システム）で運用されてきました。2015年ごろから開発環境改善の必要性に直面し、2016年ごろより試行錯誤の結果、GitLabとSlackによる開発環境が定着しました。
GitLab導入前の課題や状況としましては「会社の成長や外部環境の変化に伴う品質リスク増大」「開発部門からの問題提起により全社の開発インフラを刷新する機運の高まり」の2点が挙げられます。
会社が成長し社員が増える中で開発パフォーマンスを維持する必要性が高まりました。開発高速化の対象は時代に応じて変化しており、近年はAIやIoTなど大量データを処理する必要があります。また、プログラムの価値を確認するためのテストの量、質も変わってきました。
コードレビューやテストの重要性の高まりに伴い、社内では自前でサーバー立てて管理しはじめるチームがあるものの、チームごとにやり方が別々で、自前のサーバー管理にも課題がありました。
全社開発インフラ刷新にあたっては、開発部門、経営陣双方の要望に応える必要がありました。開発部門の要望は、バージョン管理システムをSubversionからGitへシフトすること、個別サーバー管理の負担を軽減すること、CI/CDを導入すること。一方、経営陣の要望は、全社的サービス品質のバラつきを低減すること、開発手法を統一することでした。トライアル評価の結果、現場、経営双方の要望に応えられると判断して導入が決定したのが、GitLabです。GitLab導入により実感できた効果は次の2点です。
1つめは「コードレビュー・CI/CDが定着したこと」です。イシュー、マージリクエストを軸にしたワークフローの中で、コードレビューが自然に行われるようになりました。また、CI/CDの仕組みがプロセスに組み込まれ、CI/CDを実施する文化が定着しました。
2つめは「開発部門の標準的なマネジメントシステムとして定着したこと」です。全社的なさらなるサービス品質向上のため「社内標準開発プロセス策定ワーキンググループ」を発足し、同グループ内にてGitLabを活用した開発手法の標準化などを検討し、社内への啓蒙活動を実施しました。
これらにより、当初のGitLab導入目的は十分に達成されたと評価しています。すなわち、CI/CDの実施、コードレビュー実施率の向上による品質向上や、現場のサーバ管理負担低減といったポイントです。
また、導入後に気づいた効果として、組織で統一された開発環境を利用することによる「ノウハウ伝授のしやすさ」「オンライン上でのコミュニケーションのしやすさ」といったポイントも挙げられます。特に、コロナ禍の中でもリモートワークで生産性を落とさず業績を維持できたのはGitLabの功績が大きいと考えています。

GitLabの全社業務への広がり

そして、開発部門での成功を受け、現在はソフトウェア開発以外のあらゆる部署の業務でGitLabを活用しています。たとえば「問い合わせ管理」への活用では、問い合わせフォームを通じ社外から問い合わせがあった際に、GitLabのWeb APIを活用しイシューを自動登録する仕組みを構築しました。
問い合わせ内容には引き合いやパートナーからの営業、提案などが含まれます。これらを適切な部門に振り分けることができるため、最適な対応が行えるようになりました。
また「営業パイプライン管理」の活用もユースケースの一つです。各プロジェクトでお客様からいただく案件を、ボード機能を使って管理しています。ラベル機能を用いて、営業確度ごとに振り分けて管理することで、高機能なCRMがなくてもGitLabをCRM的に使うことができることが分かりました。

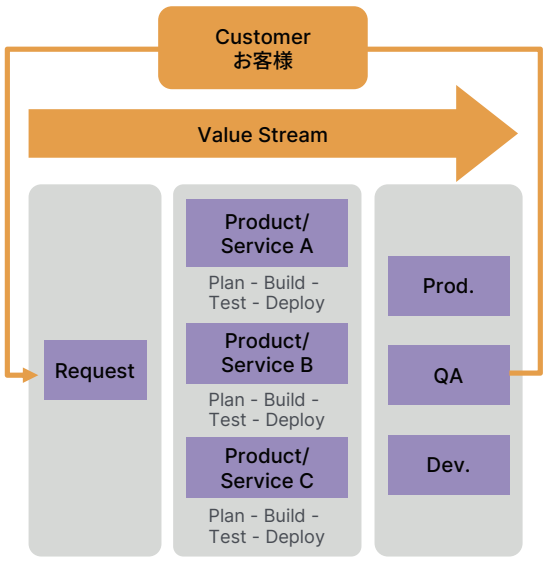
GitLabを活用している部門



そして、「IT部門のタスク管理」への活用では、社内のあらゆる業務をイシューテンプレート化し、やるべき作業をチェックボックス付きで定義して運用しています。これにより、業務の属人化を減らす効果があり、メンバー間の業務負荷のバランシングや全体納期の短縮を実現することができました。
このように、GitLabは、開発業務からの導入であったものの、全社業務で活用される汎用的な業務プラットフォームとしての活用が進んでいます。

DevOps推進プラットフォームとしてのGitLab

そして、ソフトウェア開発においては、DevOps推進プラットフォームとしての活用も進めています。
DevOpsとは、ご存知のとおり「顧客に最速で価値を届けるために、自分たちのバリューストリームを定義し、それを速く回すための一連の行為、又は方法論」と定義されます。
バリューストリームはDevOpsにおいて最も重要な概念で、「ビジネス上の仮説を立案してから、顧客に価値を送り届ける技術サービスを生み出すまでの間に必要なプロセス」のことです。
パートナー企業とのDevOpsのコラボレーション事例としては、ルネサスエレクトロニクス様の事例が挙げられます。当社はルネサスエレクトロニクスと車載ディープラーニング分野で協業し、同社の「R-Car」というSoC（システム・オン・チップ）向けにクラウド評価環境「GENESIS for R-Car」を開発・提供しています。



そして、チップのソフトウェア開発者やAIモデル開発者が、R-CarのAI性能をタイムリーに「GENESIS for R-Car」にて評価できるよう、GitLabやMLflowといった周辺ツールを活用してバリューストリームを改善しています。
これにより、従来は、要件発生から第1弾リリース、エンドユーザーフィードバック、改善版リリースに至る一連のサイクルに数ヵ月単位を要していたものが、現状は、細かい機能単位で、本番環境とそっくりの評価用環境を1クリックで立ち上げることができるため、一連のサイクルを飛躍的に短縮することができました。
DevOps推進にGitLabを活用することで、コラボレーションの質がバリューストリームのフロー改善に繋がることが実感できました。コミュニケーションミスや手作業による受け渡し等がバリューストリームのフローを悪くする要因となっています。GitLabにより、DevOpsに必要な要素が全て一つのプラットフォームに集約され、バリューストリームのフロー改善につながっているのです。
今後も開発の現場活用からパートナー企業との協業まで、GitLabによる良好なコラボレーションによってさらなる価値提供を継続していきたいと考えています。

フィックスターズについて

フィックスターズでは、コンピュータの性能を引き出す技術のスペシャリストたちが、日々高速なソフトウェアの開発を通じて、お客様の課題解決に取り組んでいます。社員の約9割がソフトウェアエンジニアで、ハードウェアの知見やアルゴリズム実装力、各産業・研究分野の知見を有している点が特徴です。
事業分野は、半導体や自動車、産業機器、生命科学、金融など多岐にわたり、組み込み高速化や量子コンピューティング、自動車向けソフトウェア開発など、大量データの高速処理を実現し、お客様の製品競争力を下支えすることに貢献しています。
たとえば、「画像処理アルゴリズム開発」では、高度な画像処理や深層学習等のアルゴリズムを開発できる人材が社内に限られているといったビジネス課題に対し、高速な画像処理需要に対して、経験豊富なエンジニアが製品開発を支援しています。
また、「AI・深層学習向け技術支援」では、組込みデバイス向けにAIモデルを軽量化したいといったビジネス課題に対し、AIを使うためのハードウェア選定や、高速な計算を実現するソフトウェア開発技術などの支援を行っています。

株式会社フィックスターズ
執行役員
マーケティング部長
小栗 伸重 氏

インターネット黎明期より、コンテンツ制作・Webエンジニア・SE職に従事。シリコンバレーの拠点と連携し、クラウド事業などの新規事業立ち上げをグローバル市場で経験。2021年より現職にて、マーケティング・広報・アライアンスを担当。

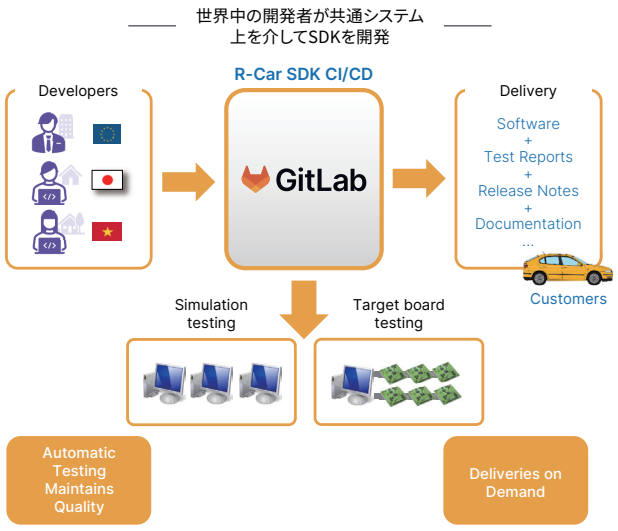
株式会社フィックスターズ
執行役員
ソリューション第二事業部長
孫 正道 氏

2010年にエンジニア入社。車載・産業機器向け組込システムのソフトウェア開発に従事。その後、管理職に転向し様々なプロジェクトマネジメントを経験。現職ではEdge AI, 深層学習, Computer Vision分野を中心に事業展開中。

リリース頻度24倍を実現
ルネサスエレクトロニクスが挑戦するCI/CD運用とは？

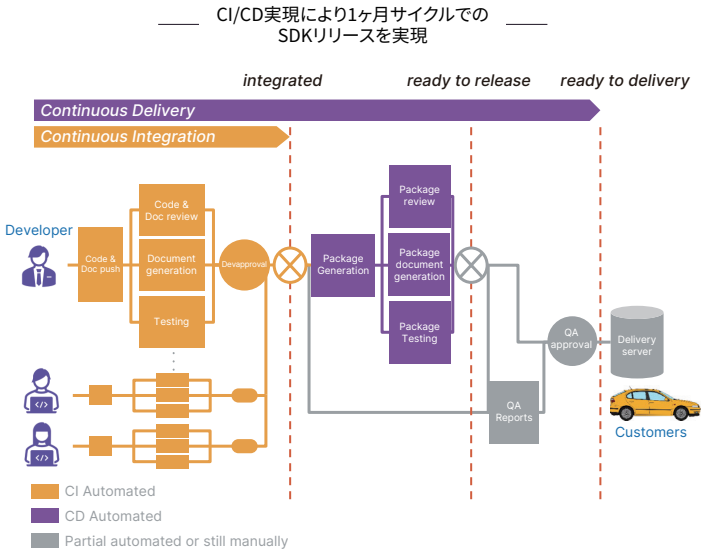
ルネサスのグローバルなCI/CDインフラ

自動車業界でなぜ、ソフトウェア開発にCI/CDが求められているのだろうか。
自動車業界は今、100年に1度の変革期と言われる。背景には「環境への配慮」「安全・セキュリティ」「快適・便利」を重視するエンドユーザーの価値の変化がある。そして、「コネクティビティ」「自動運転」「シェアード&サービス」「電動化」「パーソナライゼーション」といった業界を取り巻くトレンドが相まって、「Software-Defined Vehicle」というアーキテクチャ変革、すなわち、ソフトウェアがクルマの価値を決める時代になりつつある。
自動車のソフトウェア化によって、車載ソフトウェアは大規模化、複雑化しており、車載ソフトウェアをより早く、高品質で開発し、市場に投入できることが求められている。
こうした市場環境の中、ルネサスは「品質を損なうことなく、頻繁にソフトウェアを届ける」ことをポリシーとしてきた。しかし、ソフトウェアのリリース頻度は早くても6ヵ月、長くて2年であった。よりリリース頻度を高め、お客様へ頻繁に信頼性の高い製品を届けるために、最新のCI/CD技術による段階的な開発と完全な自動化が必要不可欠だった。



そこで、ルネサスでは、車載ソフトウェアの開発者向けに「R-Car」というSDK（ソフトウェア開発キット）を開発するCI/CDインフラを構築し、2020年からDevOpsの運用を本格的に開始した。このインフラ上で、グローバルの4つのリージョンの500人もの開発者が、共通システム上を介して30以上のプロジェクトでSDKの開発に従事している。

世界中の開発者が並行して作成したソフトウェアは、開発者がコードをプッシュするたびにシステムが自動でビルドを行い、テストされ品質が確認される。そして、SDKにパッケージされ月1回の頻度でお客様である自動車メーカーに届けられている。



CI/CDのパイプラインの大まかな流れは次のとおりだ。まず、CIフェーズでは、各開発者が自身のコードをプッシュしたら、コードやドキュメント上のレビューをマージリクエスト上で実施する。ドキュメントの生成、自動テストを経て開発チームの承認、マージが行われる。そして、CDフェーズでは、1日1回の頻度でSDKにパッケージされ、SDKのテストを実施し、品質を確認して、月1回の頻度でお客様にデリバリーされる。これにより、1ヵ月サイクルでのSDKリリースを実現している。
しかし、ここで3つの課題があった。1つめは「分断された開発環境」、2つめは「品質担保と規模の両立」、そして3つめは「組み込みソフトウェアのテスト自動化」だ。DevOps開始当初は、チームごとに異なる開発ツールを利用していたため、それが開発の効率化を妨げる課題となっていた。そこで、DevOpsプラットフォームをGitLabに統一することにした。GitLab採用の決め手は大きく2つあり、1つは「ジオレプリケーション」などのグローバルな開発体制を支える機能が搭載されていること。そして2つめは、マージリクエストのアプルーバル（承認）など、組織で品質を保つための運用に有用な機能が搭載されていることだ。

ルネサスでのGitLab活用事例

では、具体的にDevOpsにどのようにGitLabが活用されているかを3つのポイントで紹介する。
1つめのポイントは、「グローバルなCI/CDインフラ構築を支えるサポート体制」だ。ルネサスでは、グローバルで多くの開発者が関わっている。そこで、開発インフラ構築の専門チームを立ち上げ、グローバルなCI/CDインフラを構築、開発体制を支援することとした。
同チームは、GitLabのシステム構築、保守運用を担うだけでなく、自動テストの仕組みやテストに必要なランナーマシン、評価ボードの管理も担う。そして、CI/CI共通開発インフラ構築と併せ、ガイドラインやワークフローの定義も行った。構築した共通開発インフラは、まずトライアル運用で少数のチームで運用し、そこでのフィードバックを踏まえて、段階的にSDK開発に適用していった。しかし、トライアル後、開発者から「使い方がわからない」「CIジョブを作れない」などの質問や要望が届いた。
これに対し、インフラチームでは定期的にトレーニングを実施するとともに、フォードバック内容をガイド改善に反映させるなどの地道なサポートを行い、問題の解決に取り組んだ。
続いて、ある程度、開発者が共通開発インフラの使い方に習熟してくると、今度は開発者がCI/CDの文化に追随できない課題に直面することもあった。すなわち、「日々コミットされない」「テストケースが増えない」といった状況によって、月1回のリリースサイクルに合わせて開発者がコミットするケースが生まれ、リリース直前に大きなインテグレーションが発生してトラブルが起きるといったケースだ。
これに対し、GitLab Issueを活用することで、バーンダウン、バーンアップチャートでSDKリリースごとに進捗状況や課題をモニターし、改善策を実施するなどの取り組みを行い、当初はリリース時点で多くの課題が次のリリースに持ち越される状況だったものが、2ヵ月ごとにギャップが縮小し、最終的にはオープンな課題がゼロになるところまで改善が見られた。
2つめのポイントは、「共通リポジトリの課題とその解決」だ。品質担保と規模の両立に関する課題と解決である。500人の開発者が、SDK開発の30のプロジェクトに従事すると、複数のモジュールで更新が競合したり、実行のリソースが不足したりすることがある。
これに対し、モジュールをコードオーナーの機能を使って体系立て、ジョブの定義を変更したファイルに合わせてカスタマイズするといった施策で対応した。これにより、モジュールのコードオーナーが誰かが可視化され、プッシュ後には変更したファイルに連動したビルドテストが実行され、その結果はコードオーナーが承認するというフローが確立された。コミットによっては、複数のモジュールをまたいだケースが発生するが、マージリクエストとコードオーナーの承認により、品質が確認された状態でメインラインのマージが確認できるようになった。
3つめのポイントは「組み込み評価ボードの自動テスト」だ。従来の組み込み評価ボードのテストは、デバッガや評価ボードなどの多くの機材が必要で、OSやデバイス、ドライバを組み合わせてテストが必要なため手間がかかり、ときには、評価ボードが固まってしまうこともあった。
こうした課題に対し、評価ボードを遠隔操作できる仕組みを構築し、SDK開発者が、いつでも、どこでも、誰でもテストできるようにした。
組込み評価ボードのリモート制御にはPythonでシステムを構築。開発者がコードプッシュするとGitLab経由でCIジョブがGitLab Runnerへ送られる。そして、GitLab Runnerは電源ユニットにボード電源Onを指示し、イーサネット／シリアル通信で評価ボード上での自動テストを実行する。その後、テスト結果はGitLab経由で開発者へ送られるというフローが確立された。さらに、ネットワーク対応の電源制御ユニットにより、ボード不調時などの緊急時でもリモートで評価ボードの電源が管理できるようになった。

今後の展望

こうした取り組みによって、モダンなDevOps開発フローへ進化させることを実現し、結果としてSDKリリース頻度は大幅に増加した。リリース頻度は最長で2年（24ヵ月）だったものがわずか1ヵ月に退縮され、リリース速度は24倍に向上した。
今後は、ADAS（先進運転支援システム）向けだけでなく、Gateway向け、車載用MCU向けソフトウェア開発にもこの仕組みを展開することで、お客様の製品開発に貢献していきたいと考えている。
さらに、ルネサス内部の仕組みを外部展開化することも視野に入れている。これにより、評価ボードを所有したりSDKをインストールしたりしなくても、プラットフォームベースのソフトウェア開発を可能にするクラウド環境によって、フルリモート開発を実現していきたい。

ルネサスエレクトロニクスについて

ルネサスエレクトロニクスは「To Make Our Lives Easier」を掲げ、自動車、産業、インフラ、IoTの4つの成長分野へソリューションを提供することで、より安全で、健康でスマートな社会に発展させることをパーパスとしている。
半導体デバイスをはじめとするルネサスの製品は、日々の暮らしに欠かせないあらゆる組込み機器に搭載されている。特に、自動車分野においては、信頼性の高い車載制御や自動運転技術、電気自動車などに適用されている。
今回は、自動車分野のソフトウェア開発にフォーカスしてCI/CD（継続的インテグレーションと継続的デリバリー）の取り組みを紹介したい。

ルネサスエレクトロニクス株式会社
オートモーティブソリューション
事業本部
車載ソフトウェア開発統括部
主任技師
永井 優 氏

2007年ルネサスエレクトロニクス(株)に組み込みソフトウェアエンジニアとして入社。民生・車載機器向けSOCでの組み込みソフトウェア開発に従事し、幅広い製品へ貢献。現在はソフトウェアアーキテクトとして車載ソフトウェアプラットフォーム開発を推進しつつ、DevOpsを活用した組織生産性の向上にも取り組んでいる。


GitLab