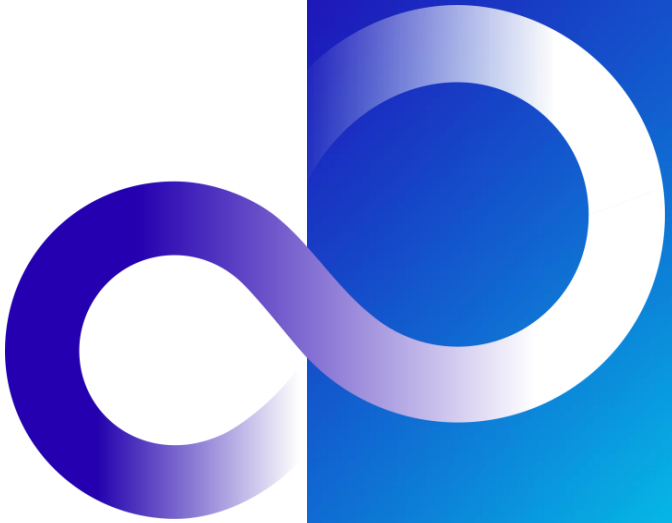
FUJITSU Hybrid IT Service Digital Application Platform「DevOps with GitLab」「GitLab EE Subscription」ご紹介資料

2023年9月

富士通株式会社



・本資料の無断複製、転載を禁じます。
・本資料は予告なく内容を変更する場合がございます。
・本資料には富士通クラウドテクノロジーズ株式会社が制作したコンテンツが含まれます。

Version 2.0

目次

※ 本資料では、「FUJITSU Hybrid IT Service FJcloud-V」を「FJcloud-V」、「FUJITSU Hybrid IT Service Digital Application Platform」を「Digital Application Platform」と記載する場合があります。
「FJcloud-V」、「Digital Application Platformの一部のサービス」は富士通クラウドテクノロジーズ株式会社が提供するクラウドサービス「ニフクラ」により、クラウド環境の運用を意識することなくVMwareベースの各種機能を利用可能なサービスです。

1. DevOpsとDevOpsの導入障壁

1.1 DevOpsとは

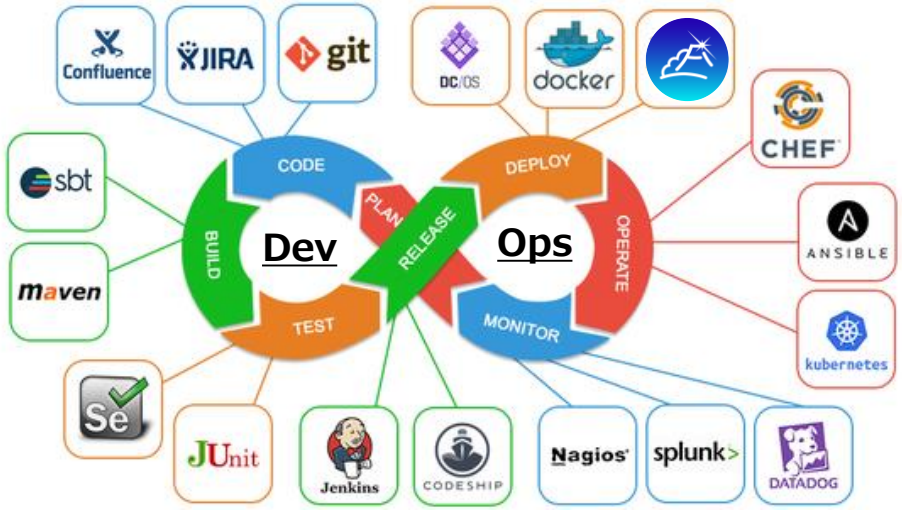
- DevOps(デブオプス)とは、ソフトウェア開発の手法のひとつで、**開発チーム(Dev**elopment**)と運用チーム(Op**erations**)が協力しあってシステムを開発・運用することで**、ビジネスの価値を高めるための様々な取り組みを示す概念です。
- 従来のプロセスと比較して、ソフトウェアの開発と配信の効率、速度、セキュリティを向上させます。より機敏なソフトウェア開発ライフサイクルは、企業とその顧客に競争上の優位性をもたらします。

組織を超えた、協力し合う「文化醸成」と「ツールの活用」が重要

FUJITSU

DevOpsの導入コスト増加など課題が発生

- ステップごとに使用するツールの種類が多く、**どこから手を付けていいか分からない**。
- DevOpsの体系的な知識がまとまっておらず、**DevOpsの進め方が分からない**。
- チームごとに精通しているツールが異なるため、**チーム間の連携時に問題が発生すること**がある。



DevOpsの各フェーズにおけるツール群の例

これら全てのツールを習得し使いこなすのは
非常に困難

DevOpsを統合的に利用でき、
習得しやすいプラットフォームが
求められている

DevOpsの全ライフサイクルを
統合的にカバーする「GitLab」を提供

FUJITSU

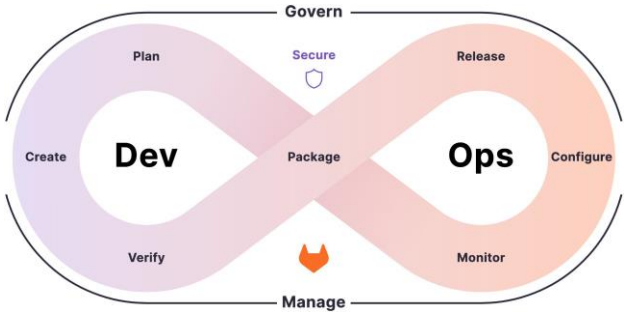
2. GitLabとは

2.1 GitLabとは

GitLabは組織全体（ソフトウェア、運用、IT、セキュリティ、ビジネス）に対応できる、完全な透明性、一貫性、トレーサビリティを持つエンドツーエンドの統合システムであり、ソフトウェアの共同計画、ビルド、防御、デプロイを可能にするために設計されたDevSecOpsプラットフォームです。

単一ツールでDevSecOpsすべてのライフサイクルを機能として提供

- 1つのUIでプロジェクトの開発・管理・運用をカバー
- 日本の商習慣に相応しい権限設定
- セキュリティチェックなどのセキュリティ機能
- プロジェクトのビジネスROI効果を測定可能



機能概要一覧

計画	作成	検証	パッケージ	セキュリティ	リリース	構成、設定	モニター	防御	管理
チーム計画	ソースコード管理	継続的インテグレーション (CI)	パッケージレジストリ	SAST	継続的デリバリー	自動DevOps	指標	コンテナスキャン	サブグループ
ポートフォリオ管理	コードレビュー	コードテストとカバレッジ	コンテナレジストリ	秘密の検出	ページ	Kubernetes管理	事故管理	セキュリティオーケストレーション	イベントの監査
サービスデスク	ウィキ	性能試験	Helm Chartレジストリ	コード品質	高度な展開	展開管理	オンコールスケジュール管理		監査レポート
要件管理	Web IDE	マージトレイン	依存関係プロキシ	DAST	機能フラグ	ChatOps	トレース		コンプライアンス管理
品質管理	スニペット	アプリを確認する	Git LFS	APIセキュリティ	証拠を公開する	コードとしてのインフラストラクチャ	エラー追跡		DevOpsレポート
デザインマネジメント		シークレットマネジメント		ファジングテスト	オーケストレーションをリリース		製品分析		バリューストリーム管理
				依存関係のスキャン	環境管理				
				ライセンスコンプライアンス					
				脆弱性管理					

画像引用元:
- https://about.gitlab.com/topics/devops/
- https://about.gitlab.com/why-gitlab

2.1 GitLabでのDevOpsの仕組み

FUJITSU

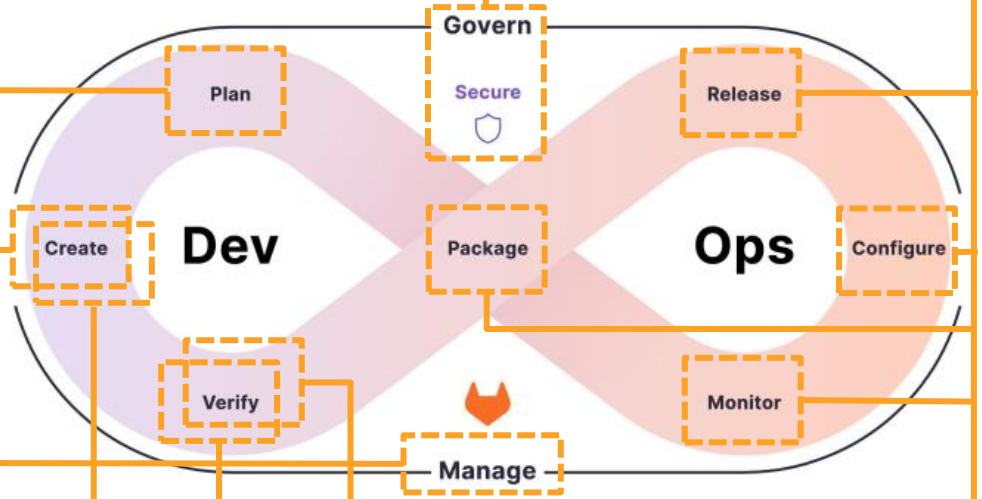
2.2 ポートフォリオ、プロジェクトマネジメント

- グループ
- ロードマップ
- エピック
- プロジェクト
- マイルストーン　マイルストーン

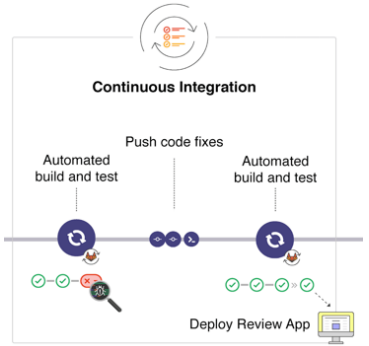
Issue　などPM機能

2.5 セキュリティ

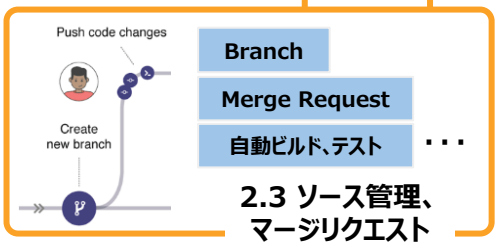
App Sec　SAST　DAST　...



CI/CD パイプライン

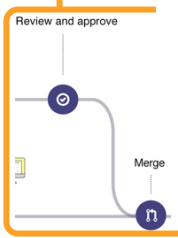


- Runner
- Pipeline
- Env管理
- IaC
- K8s Monitor
- ...



- Branch
- Merge Request
- 自動ビルド、テスト
- ...

2.3 ソース管理、マージリクエスト

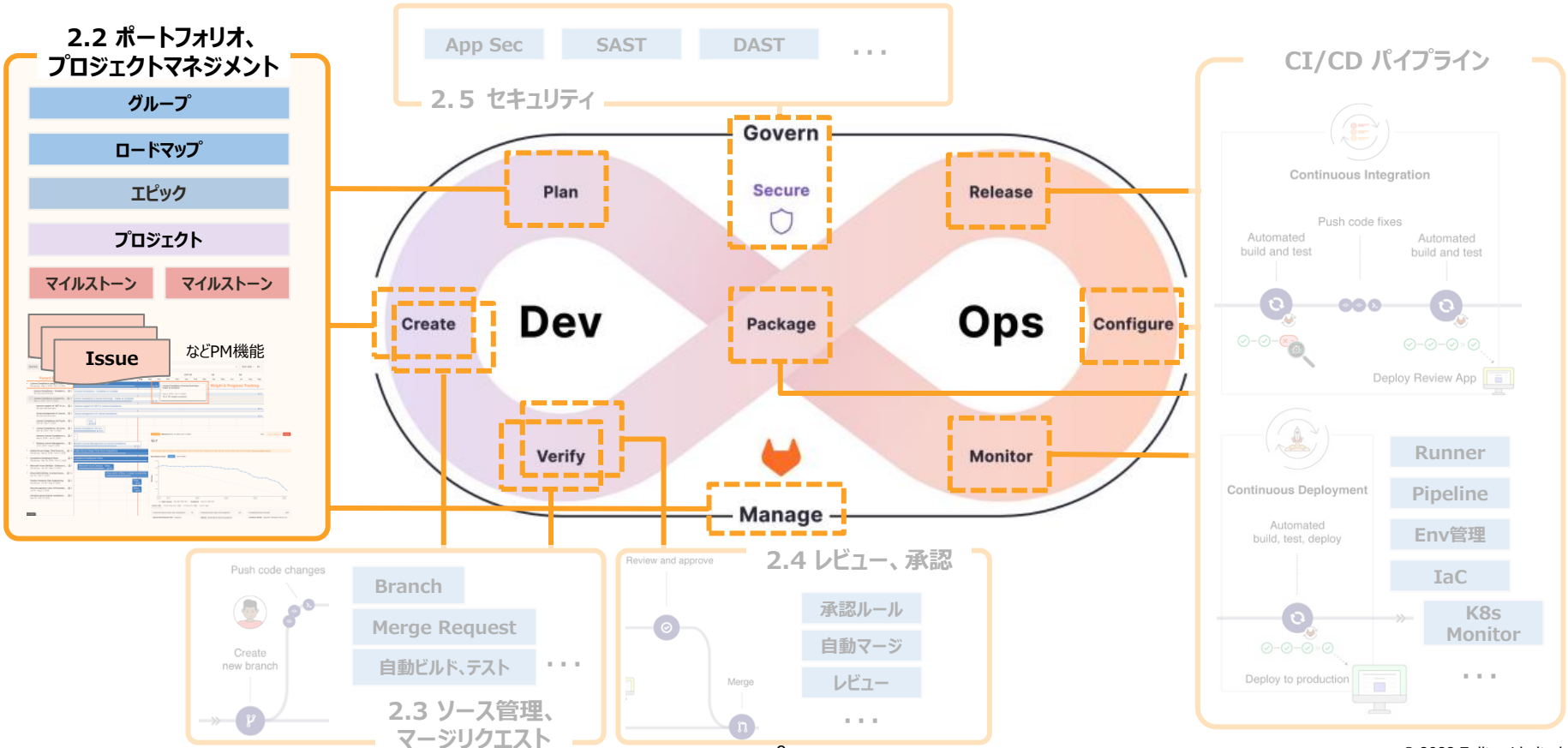


2.4 レビュー、承認

- 承認ルール
- 自動マージ
- レビュー
- ...

2.2 ポートフォリオ、プロジェクトマネジメント機能

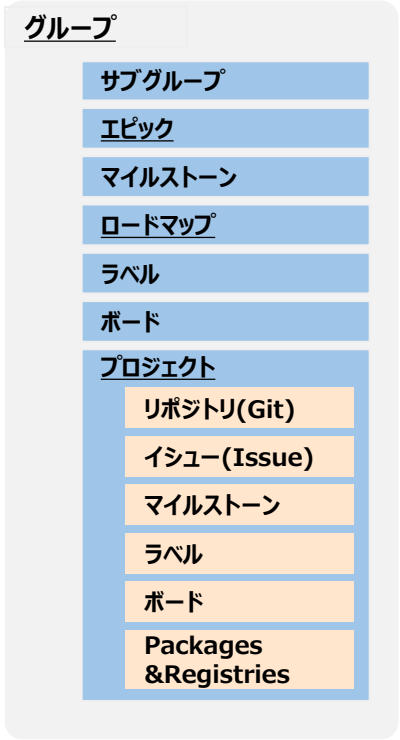
FUJITSU



2.2 ポートフォリオ、プロジェクトマネジメント機能

FUJITSU

○ 複数階層、複数チームを持つ組織におけるプロジェクト管理の効率化



グループ:

- ○ポートフォリオや、プログラムとして複数のプロジェクトを保有
- ○戦略的な計画、統制、管理をするためのレイヤーを提供

エピック:

- ○機能、開発テーマ、大きなユーザストーリー、ビジネス戦略
- ○グループに属し、子エピックとイシューを集約、WBSを実現

ロードマップ

- ○エピックの時系列的な可視化、長期的な計画のスケジュール化

プロジェクト

- ○チームがコラボレーション、計画、コーディング、アプリケーションのデリバリを実施する起点

イシュー(Issue)

- ○ユーザーストーリー、バグ、非機能要件、タスク

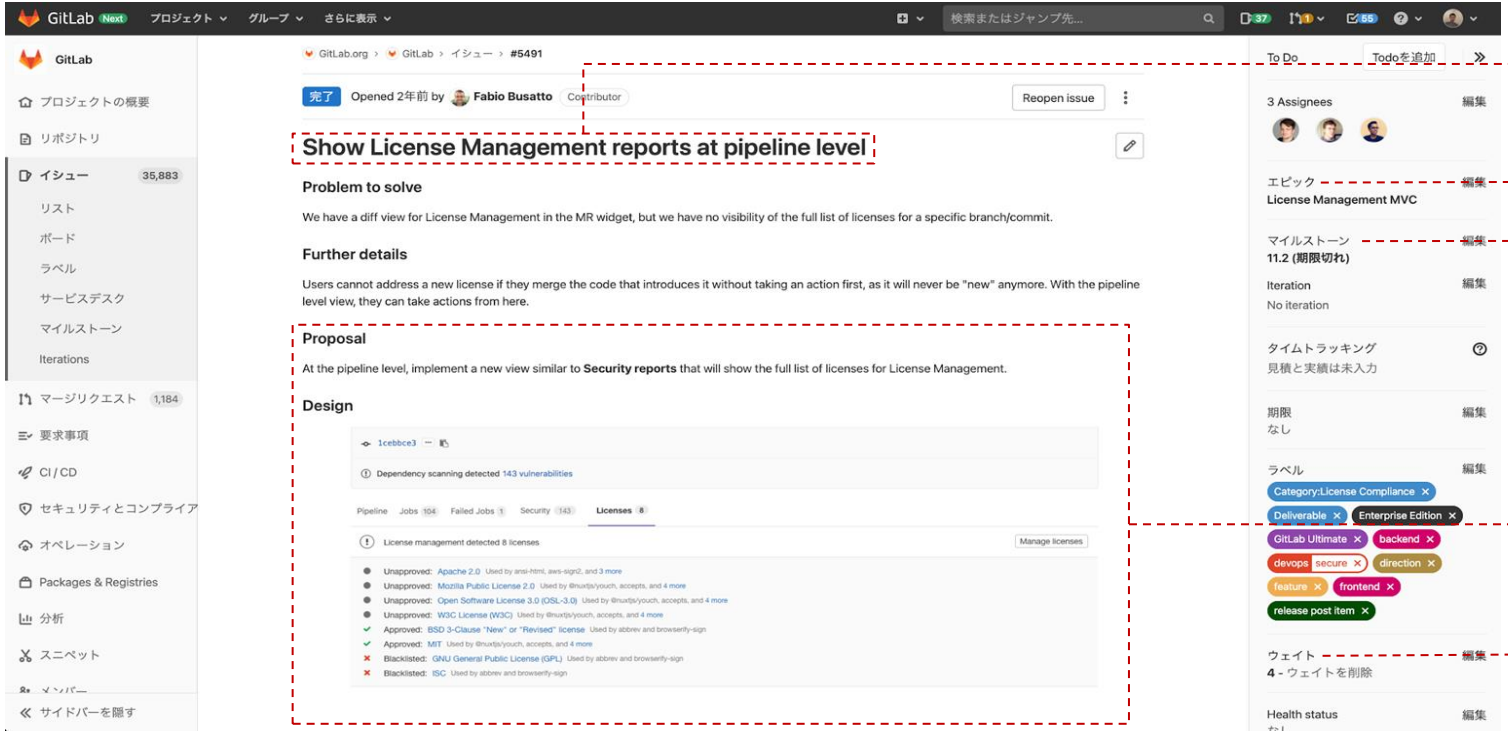
マイルストーン

- ○リリース可能成果物、イシューにもとづく固定期間の計画
- ○グループ単位、プロジェクト単位双方設定可能

2.2.1 イシュー：ユーザーストーリー、バグ、要件の定義

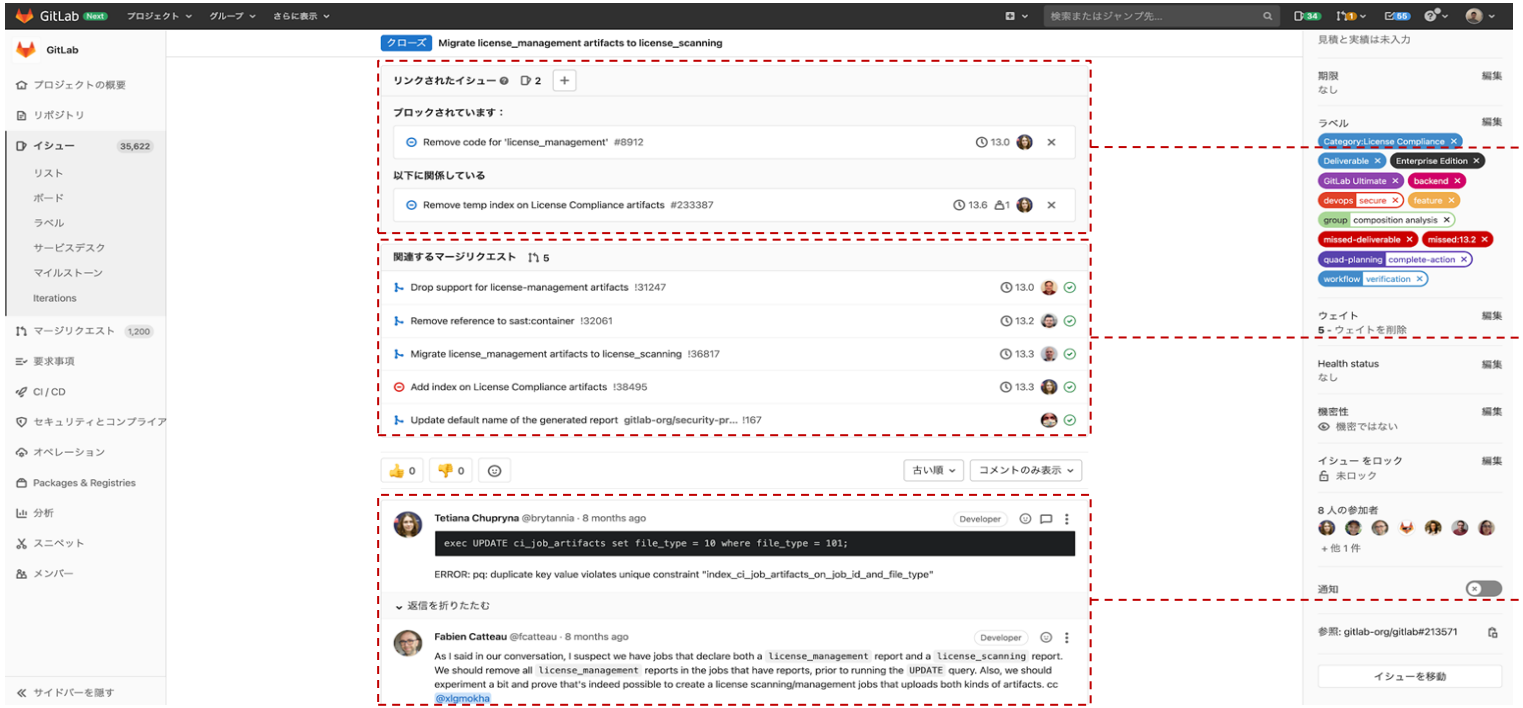
FUJITSU

○ イシューでユーザーの要件や、システムのバグなど様々なプロジェクト課題を管理できます



2.2.2 イシュー：関連、開発追跡、コラボレーション

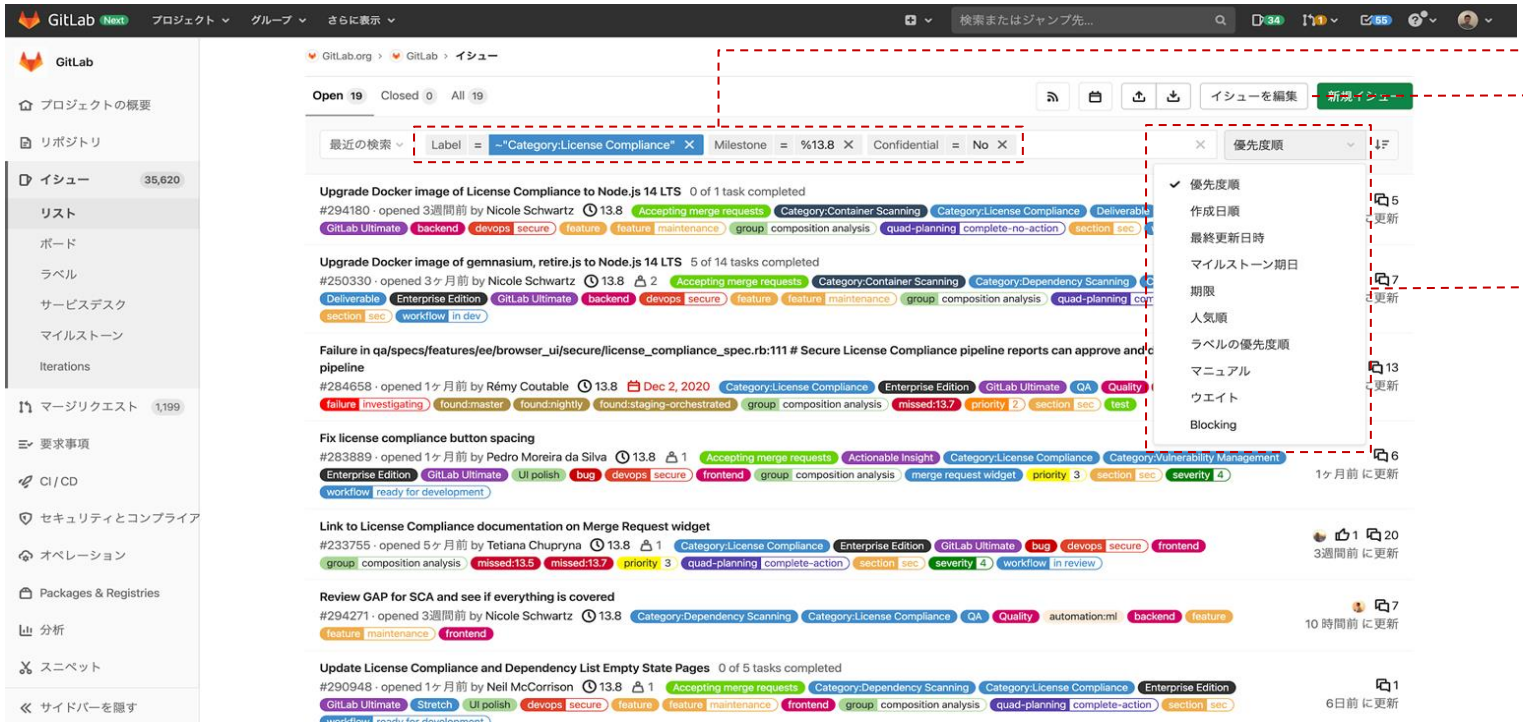
○ イシューの中に、開発者の作業記録が連携され、開発の追跡も簡単にできます



2.2.3 イシューリスト：バックログ管理

FUJITSU

○ Queryを使って、必要な条件に応じてイシューを一覧表示することができます

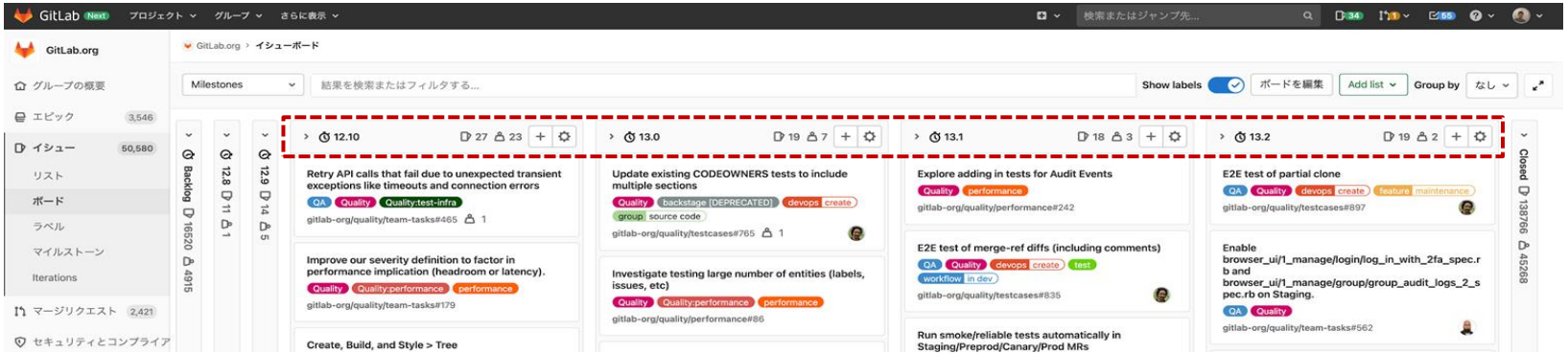


2.2.4 イシューボード：簡単に課題を分類し、操作することが可能

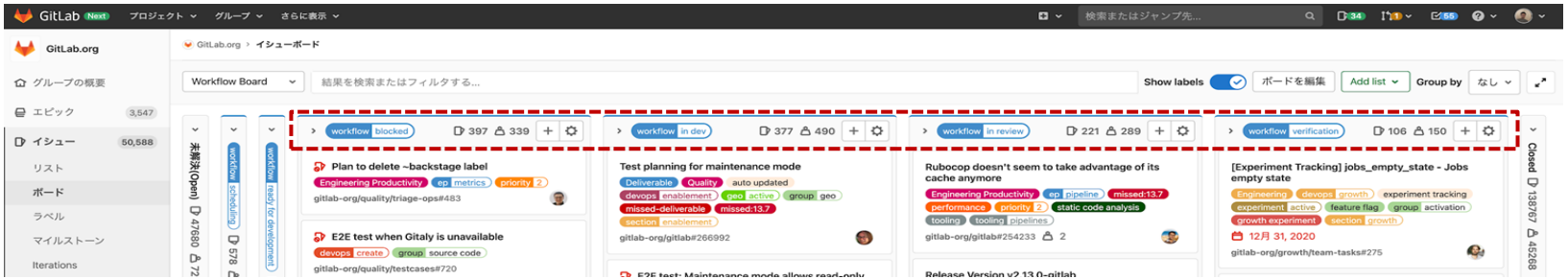
FUJITSU

○ ボード機能でユーザーが独自の条件に基づいてイシューを分類し、一括管理することができます

例:マイルストーン別 イシューリスト

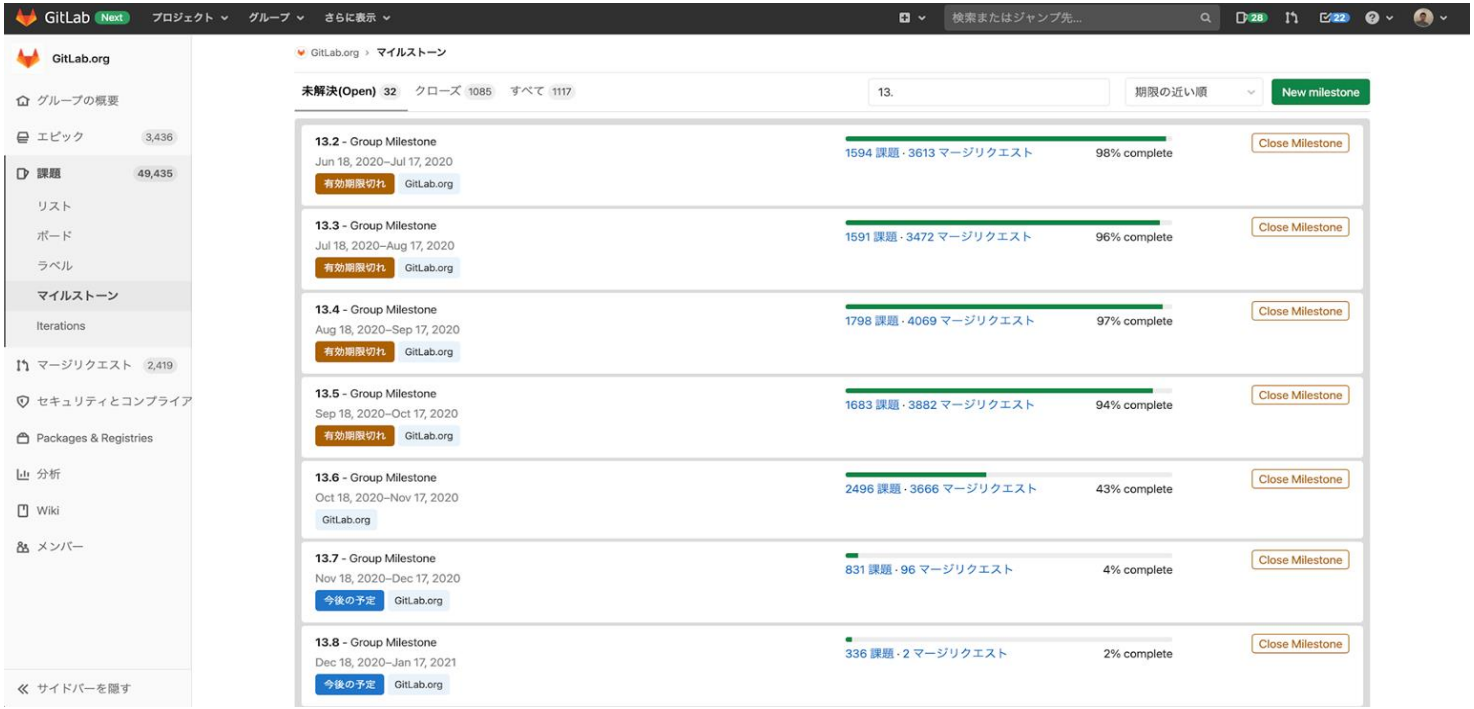


例:Label別で イシューリスト



2.2.5 マイルストーン：リリース計画

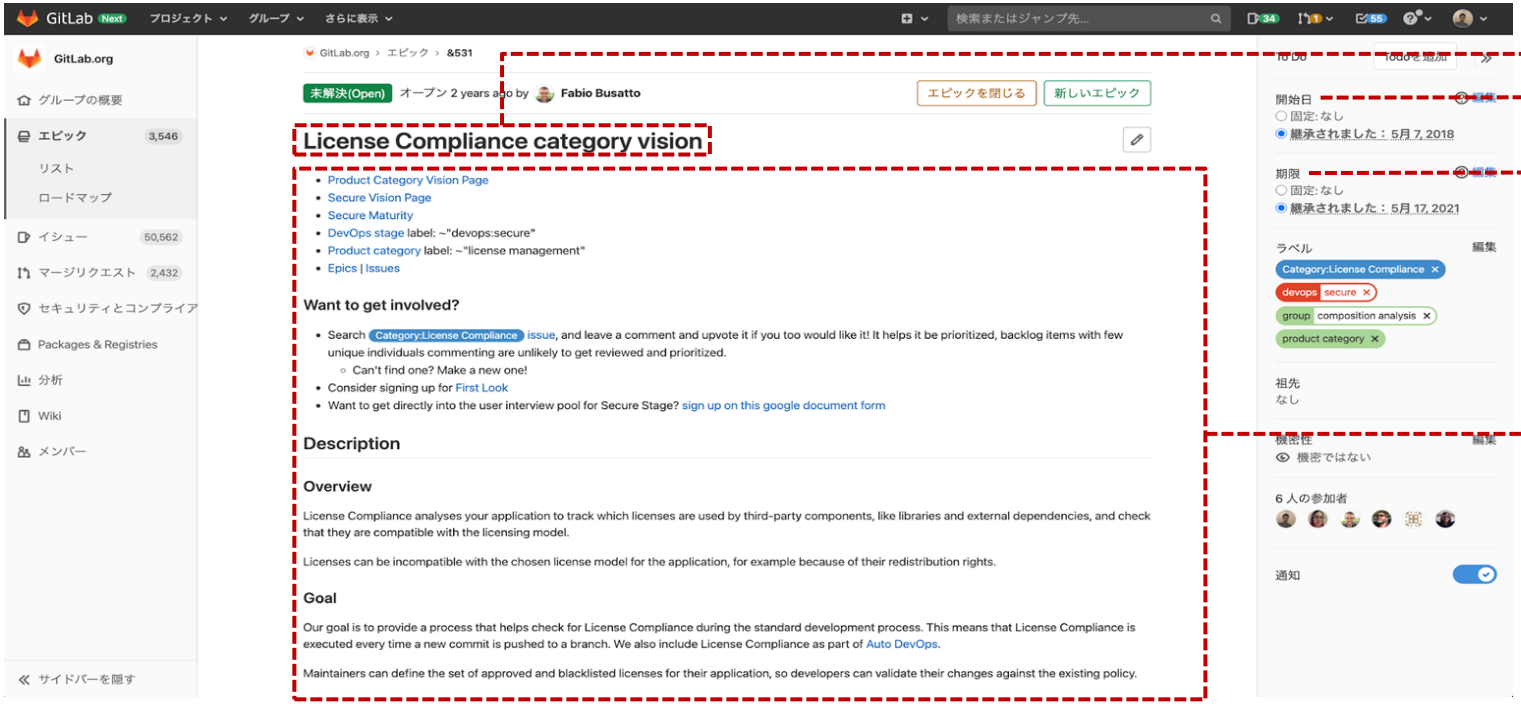
○ マイルストーン機能でリリース計画を管理することができます



2.2.6 エピック：大きなタスクの定義（イシューの集約）

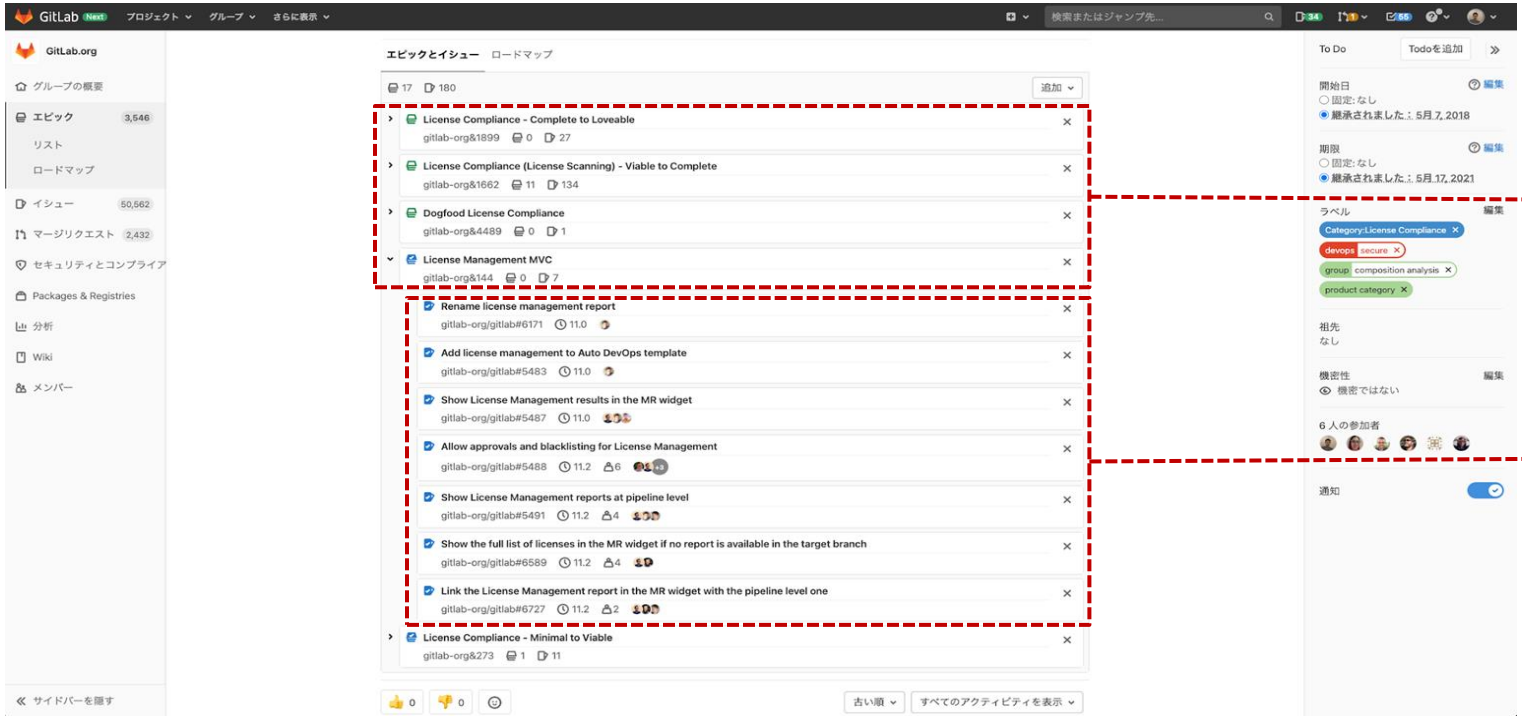
FUJITSU

○ 同じ取り組みに関連する複数のイシューを束ねて、エピックとして一括で管理することができます



2.2.7 エピック：子エピックとイシューへのWBS化

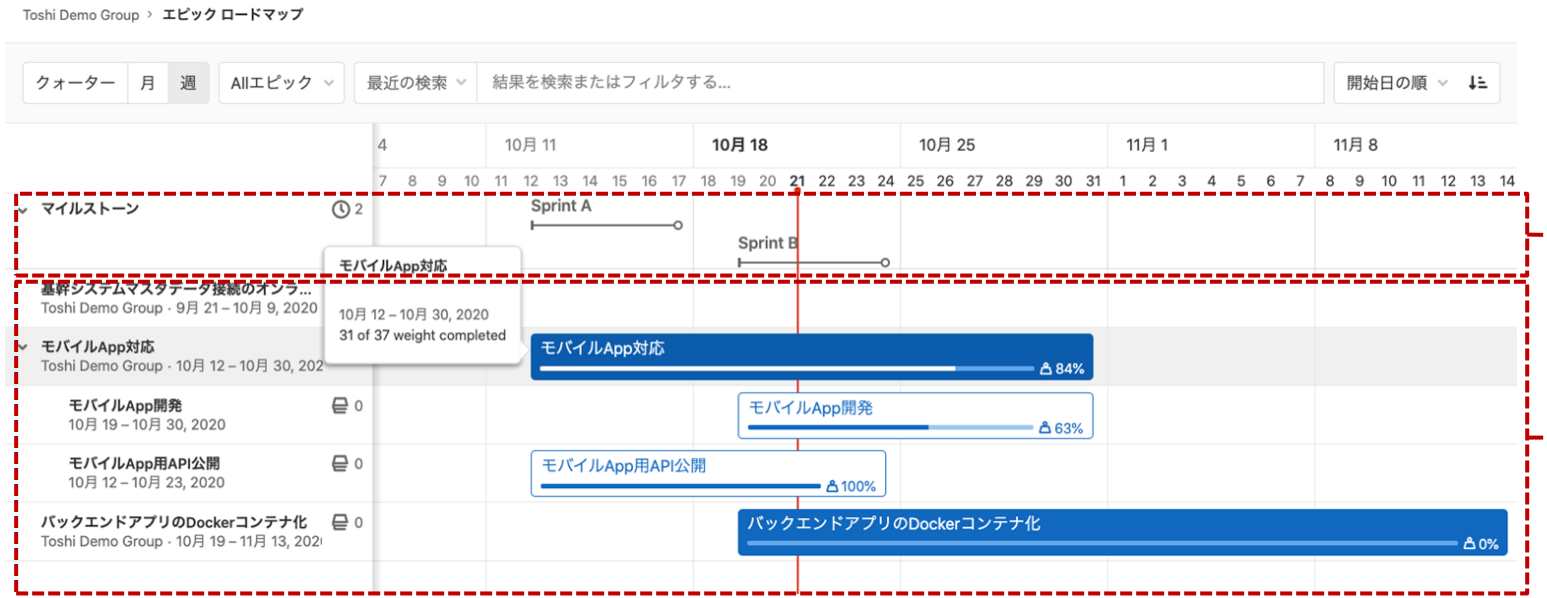
○ エピックの配下にサブエピックなど複数階層の管理ができます



2.2.8 ロードマップ：スケジュール管理、エピックの進捗可視化

FUJITSU

○ ロードマップ機能を使用すると、スケジュールやEpicの進捗状況を時系列で可視化することができます

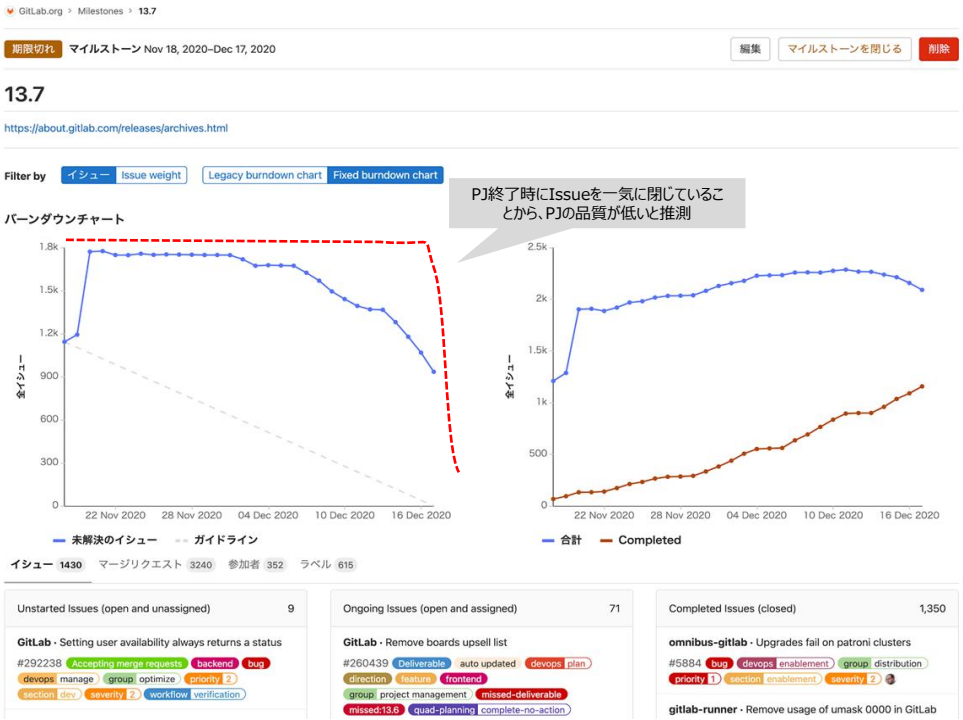


https://gitlab.com/groups/gitlab-org/-/roadmap?state=all&sort=start_date_asc&layout=WEEKS&progress=WEIGHT&show_progress=true&show_milestones=true&milestones_type=ALL

2.2.9 バーンダウン・バーンアップチャート: プロジェクト健全性可視化

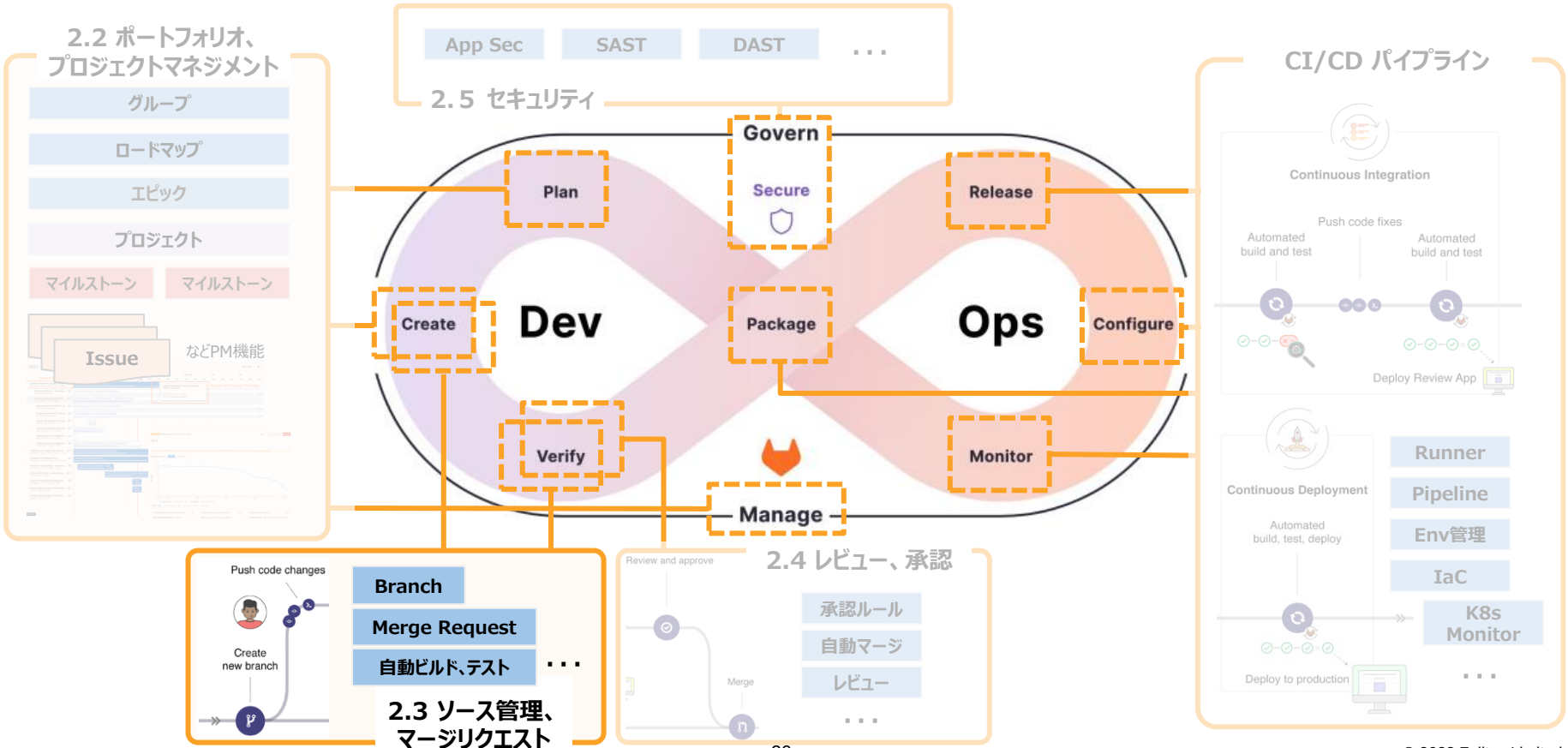
FUJITSU

- バーンダウン・バーンアップチャート機能を使うことで、プロジェクトのイシュー解決状況を時系列で表し、プロジェクトの健全性を反映することができます



2.3 ソースコードマネージメント、マージリクエスト機能

FUJITSU



2.3.1 コンカレントDevOpsの必需品：マージリクエストとは？

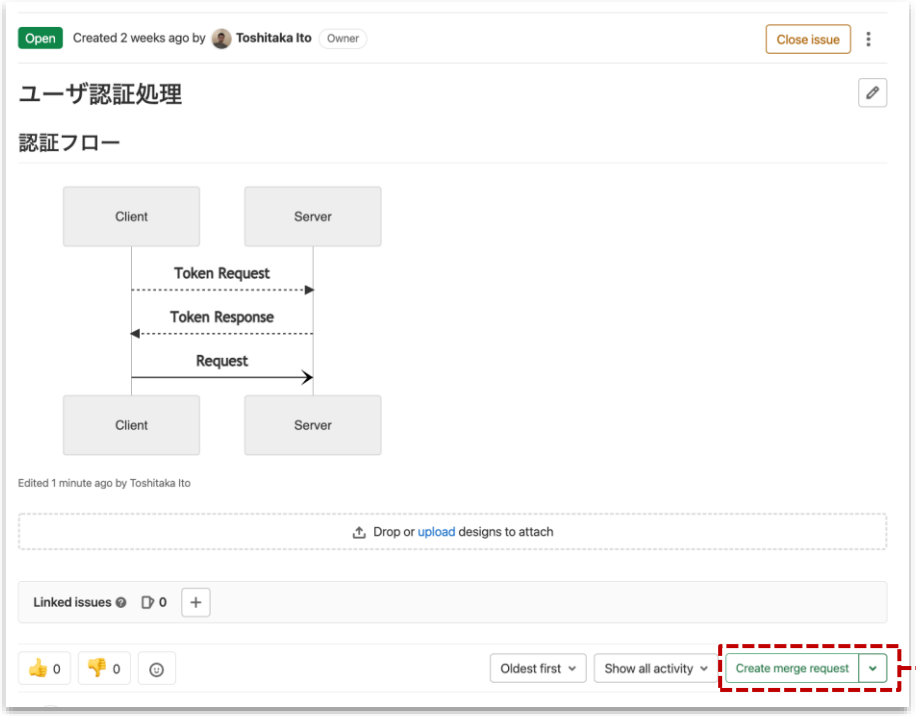
FUJITSU

- **あるブランチから他のブランチへマージを依頼するためのオブジェクト**
  - マージ元（ソースブランチ）とマージ先（ターゲットブランチ）が必要
  - ソースブランチとマージリクエストは1対1の関係
- **対話的な開発のコラボレーションをするための場**
  - Web IDEによるソースコードの修正
  - ソースブランチに対するCI/CDパイプラインの状況確認
  - レビューアーのアサイン
  - ソースコードの差分を可視化
  - ソースコード修正についてコメントや議論が可能
  - 正しい状態のソースブランチをマージするための承認プロセス
  - マージ後のイシューの自動クローズ

2.3.2 マージリクエストとイシュー(課題)の自動連携

FUJITSU

○ イシューからマージリクエストを作成可能



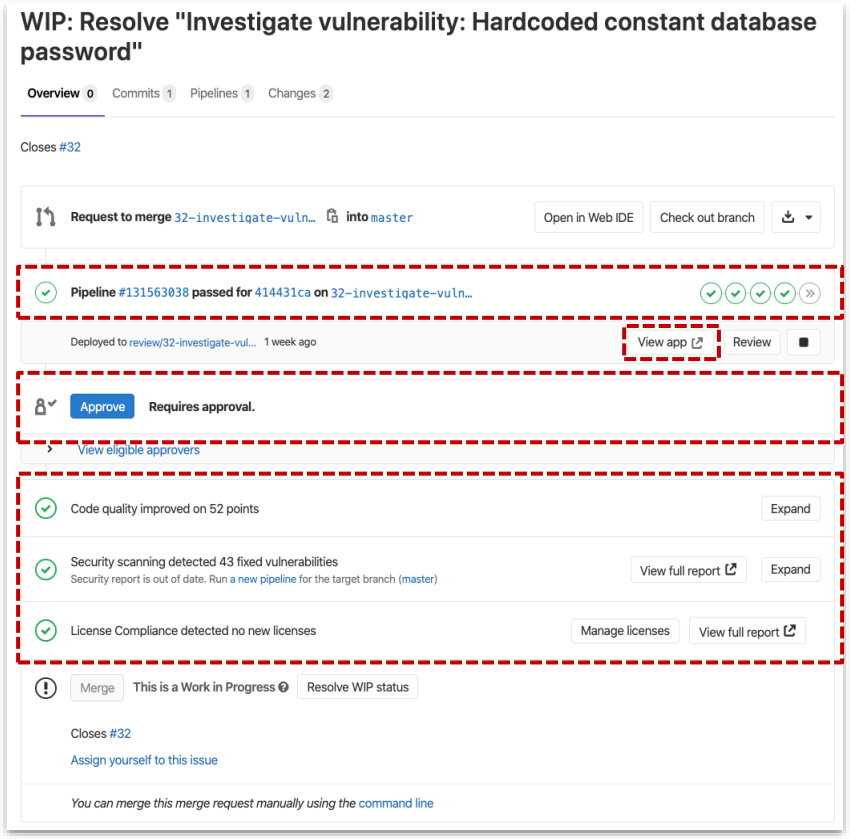
イシューから作成する利点：

- トレーサビリティ：イシューとの関連づけ、ソースブランチの同時作成によって、開発プロセスを追跡しやすくなります。
- 非開発者も含めた他のメンバーと早期に開発の状況を共有し、ディスカッションを行うことができます。
- CI/CD パイプラインの早期実行によって、早期に問題を発見し、修正することができます。

マージリクエストとブランチを同時に作成

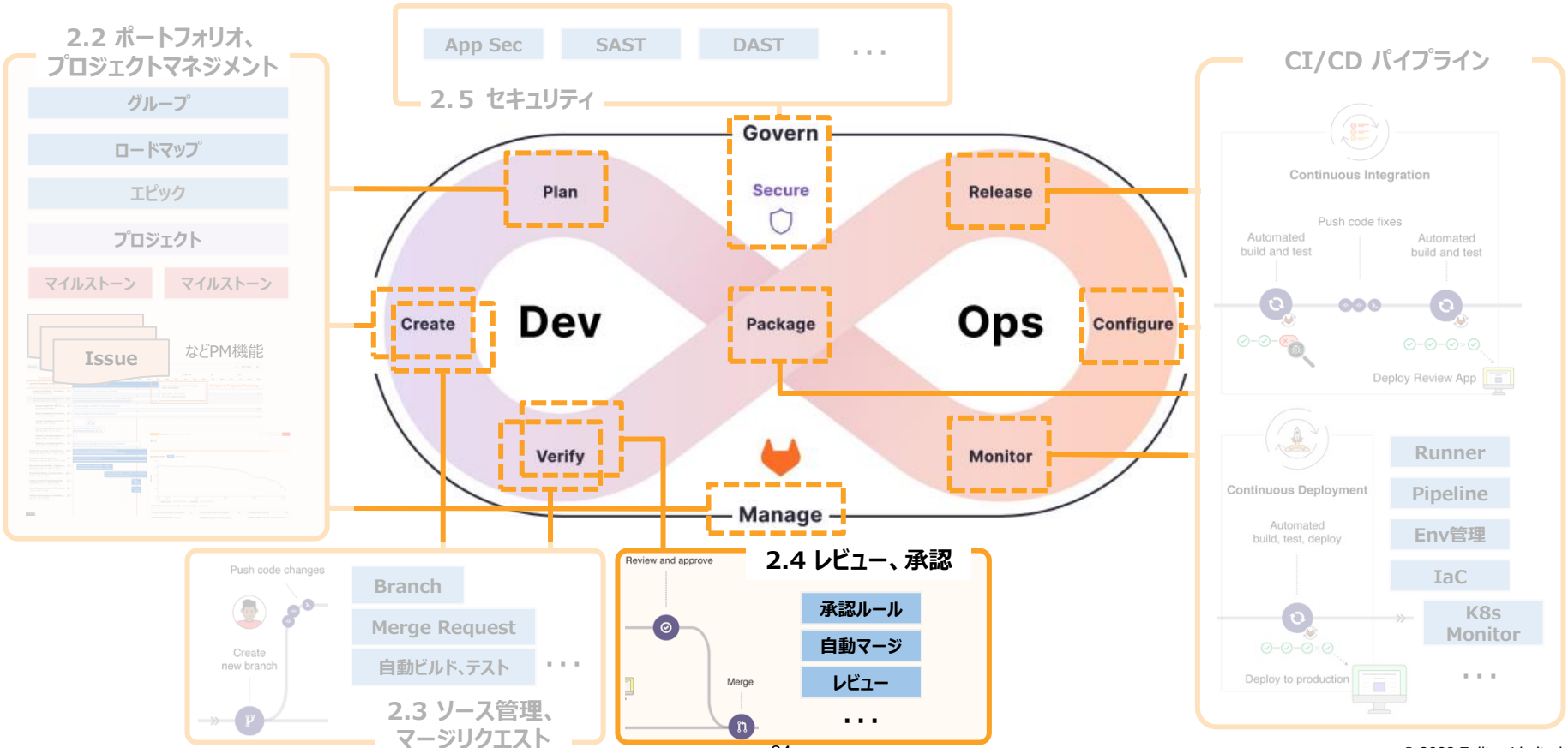
2.3.3 マージリクエストによるトレーサビリティ

- **マージリクエストにレビューが必要で、様々な情報と連携されるため、レビュー者の判断が容易になります**
  - CI/CD パイプライン 状況
  - レビュー用アプリへのリンク
  - 承認プロセス
  - スキャン結果
    - コード品質
    - セキュリティスキャン
    - ライセンスコンプライアンス
    - パフォーマンス
  - コードレビュー
  - コミット履歴
  - ソースコードの変更
  - コメント



2.4 レビュー、承認

FUJITSU



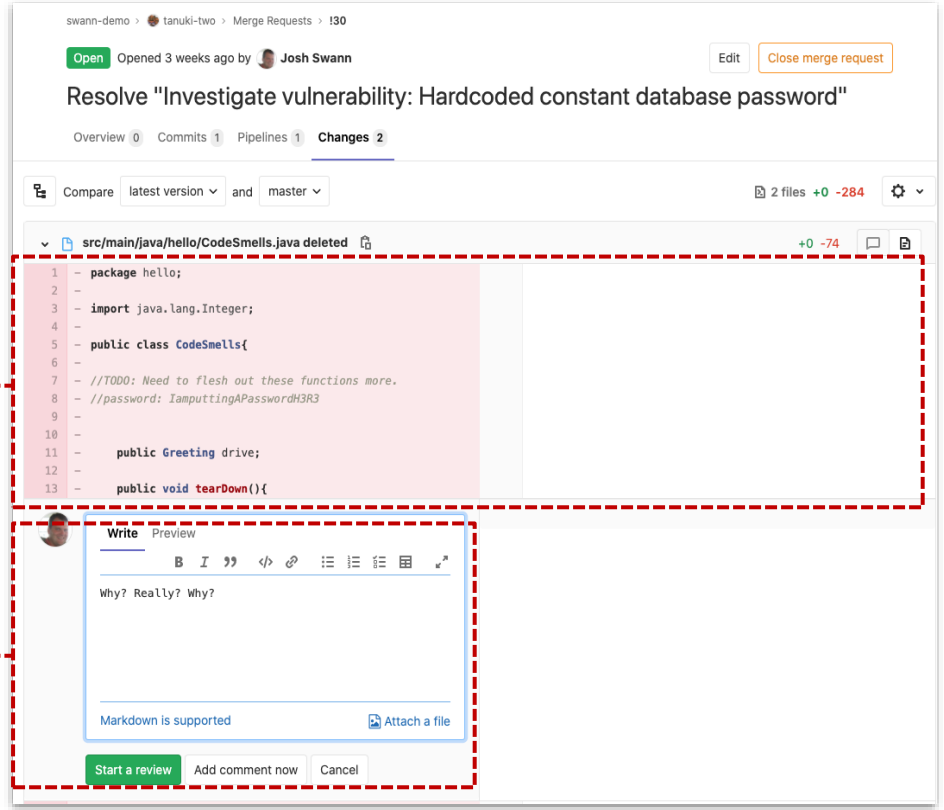
2.4.1 コードレビュー

FUJITSU

○ コードレビュー画面に色付けして、変更前後の状況を一目で確認できます

変更箇所を自動的に色分け

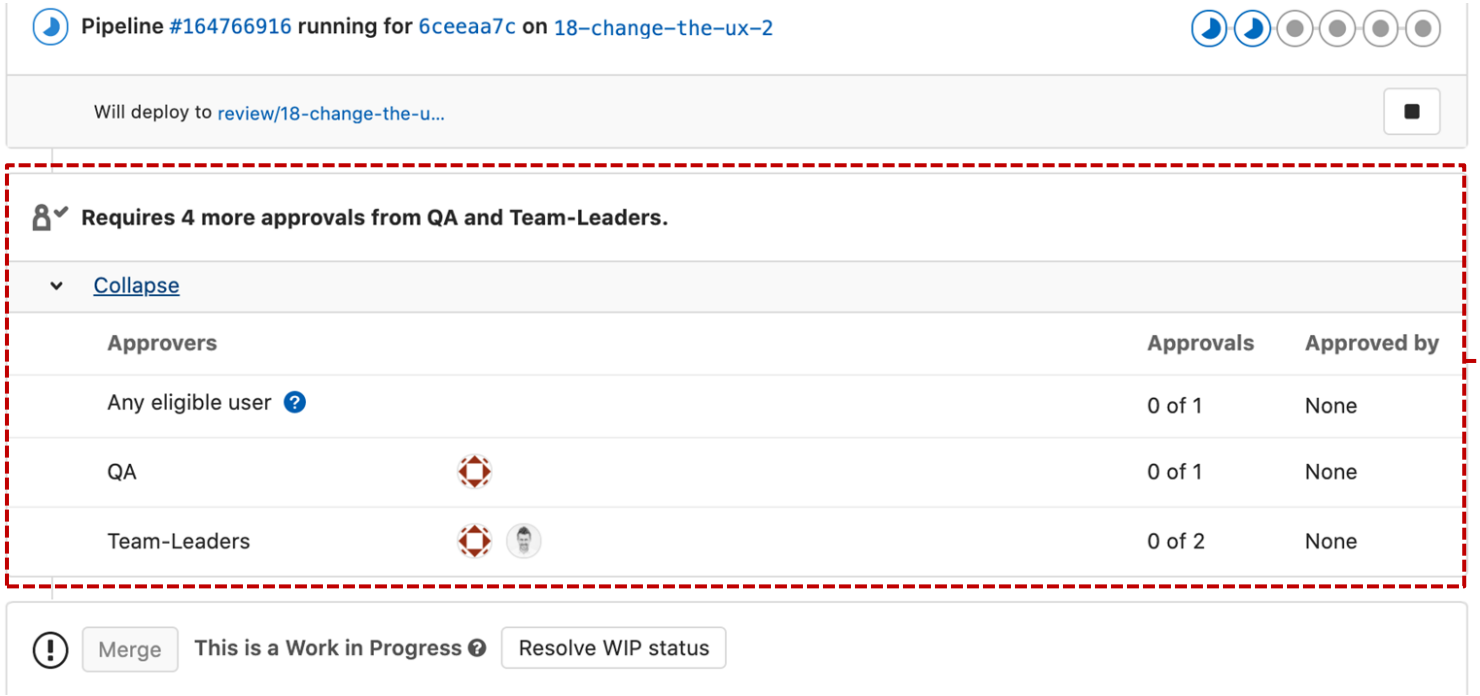
変更点についての議論を行うことができます。



2.4.1 マージリクエストの特定のユーザ/グループによる承認

FUJITSU

○ カスタマイズ可能な承認フロー機能があります

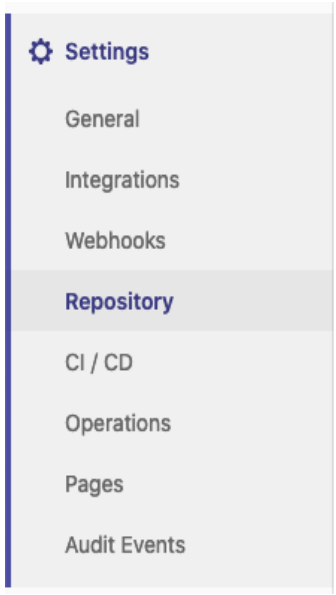


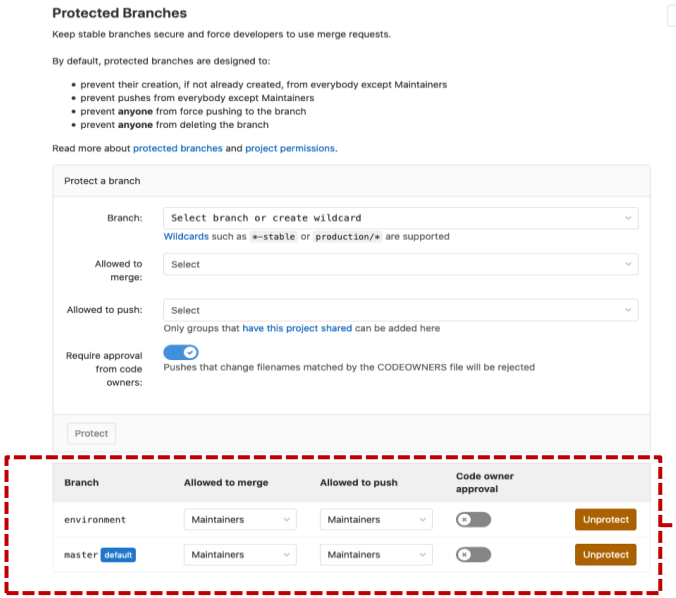
承認者と承認数のカスタマイズ

2.4.2 Protected Branches

○「Protected Branches機能」は、大事なソースコードを保護する機能です

ブランチを保護することで、コード変更の前に必要な承認やルールが遵守されているかを確認することができます。また、Protected Branches機能により、誤ってブランチを削除したり、重要なブランチに無関係な変更を行うことを防止することができます。





Master Branchや本番環境Branchへの更新は、Merge Request経由のみで行うことができ、未検証または未承認の変更は自動的に除外されます。

2.4.3 マージリクエスト承認

○ 大規模な開発において重要なブランチを保護し、マージのプロセスを体系化できます

- ○ 複数グループからの承認が必須となる
- ○ ブランチごとに承認ルールを設定可能
- ○ 特定のディレクトリやファイルに対する承認者を設定可能（Code Owner）

例：以下のようにブランチごとに複数の必要承認ルールを設定することで、マージされるコードの品質の向上と柔軟性を両立

- ○ Featureブランチ：アプリチームの任意の2名
- ○ Masterブランチ：アプリチームのリーダ + 標準化チームのメーバ任意の2名 + Code Owner

2.4.4 承認の権限付与

<承認者の例>

開発チーム	開発者		S_Dosan
			M_Motonari
	責任者		Oda_Nobunaga
			T_Hideyoshi
運用チーム	運用者		A_Mitsuhide
			D_Masamune
	責任者		T_Ieyasu
			U_Kenshin

承認ルール

名前	承認者	ターゲットブランチ	Approvals required	
資格のあるユーザー		任意ブランチ	0	
開発チーム		master	1	編集
開発チーム（責任者）		master	1	編集
運用チーム（運用者）		master	1	編集
運用チーム（責任者）		master	1	編集

Group 設定にすることも可能（上記はユーザー設定を使用）

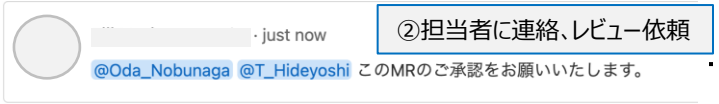
承認設定

- ユーザーがマージ リクエストで MR 承認ルールを変更できないようにします。
- 新しいコミットが MR に追加されるときに、新しい承認を要求します。
- 著者による MR の承認を防ぎます。
- MR にコミットするユーザーからの MR 承認を防止します。
- 承認のためにユーザー パスワードを要求します。

変更を保存

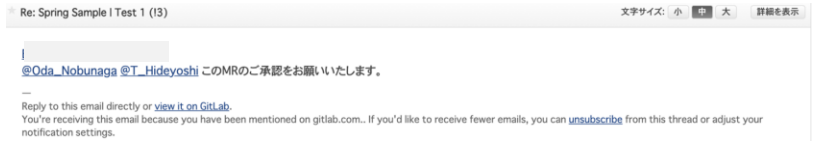
2.4.5 承認設定およびブランチ保護を実施したマージリクエスト 例

①マージリクエスト作成、承認フロー設定

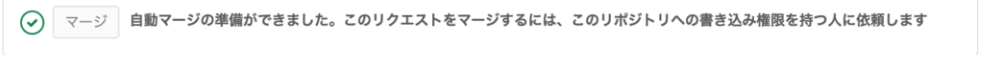


②担当者に連絡、レビュー依頼

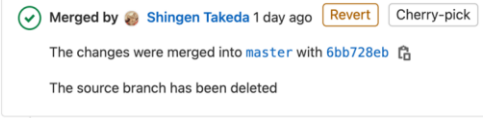
メール通知



③承認フローの通り、各担当者承認

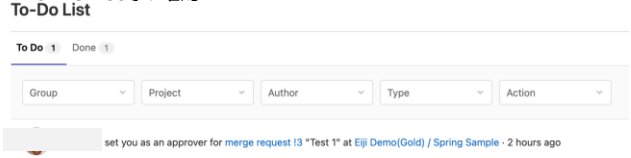


Shingen Takedaによってマージ。
（これの履歴はShingen Takedaのアカウントが削除されない限り削除されません）



④マージ条件を満たすと、マージが可能になり、変更が反映されます。

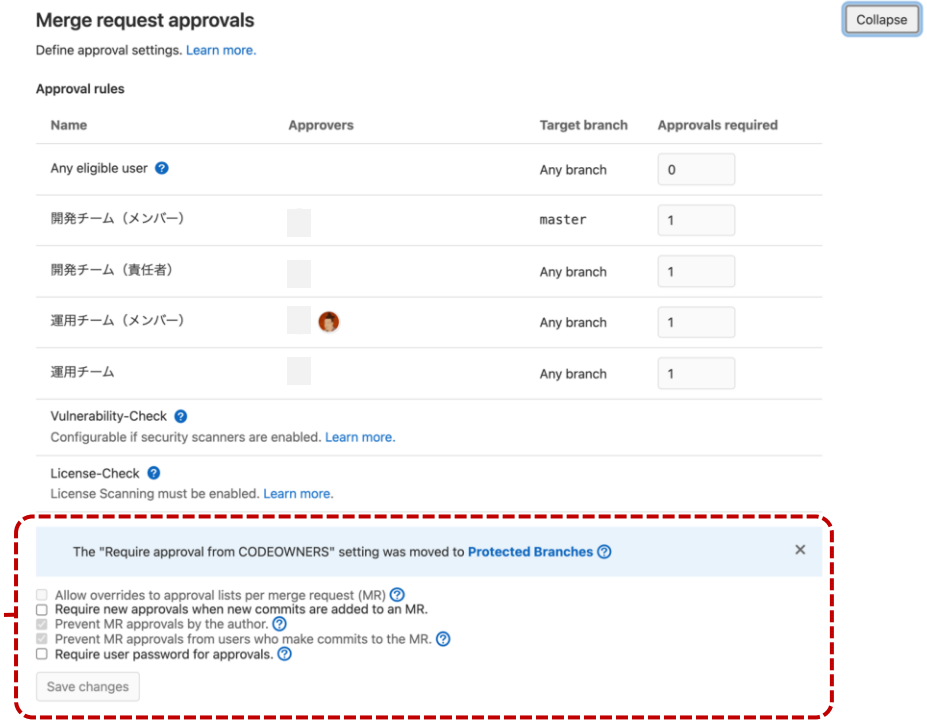
<To DO listでの確認>



2.4.6 Approval Ruleの設定

○ GitLab全体で一括の承認ルールを設定することも可能

- ○ マージリクエスト作成者による、承認の防止
- ○ MRでのコミットユーザーによる承認の防止
- ○ ユーザーによる承認ルール変更の防止



承認ルールの追加、変更が不可となっていることがご確認可能

2.4.7 Audit Event(監査イベント)

○ 下記の監査情報を提供し、監査対応が簡単に対応可能

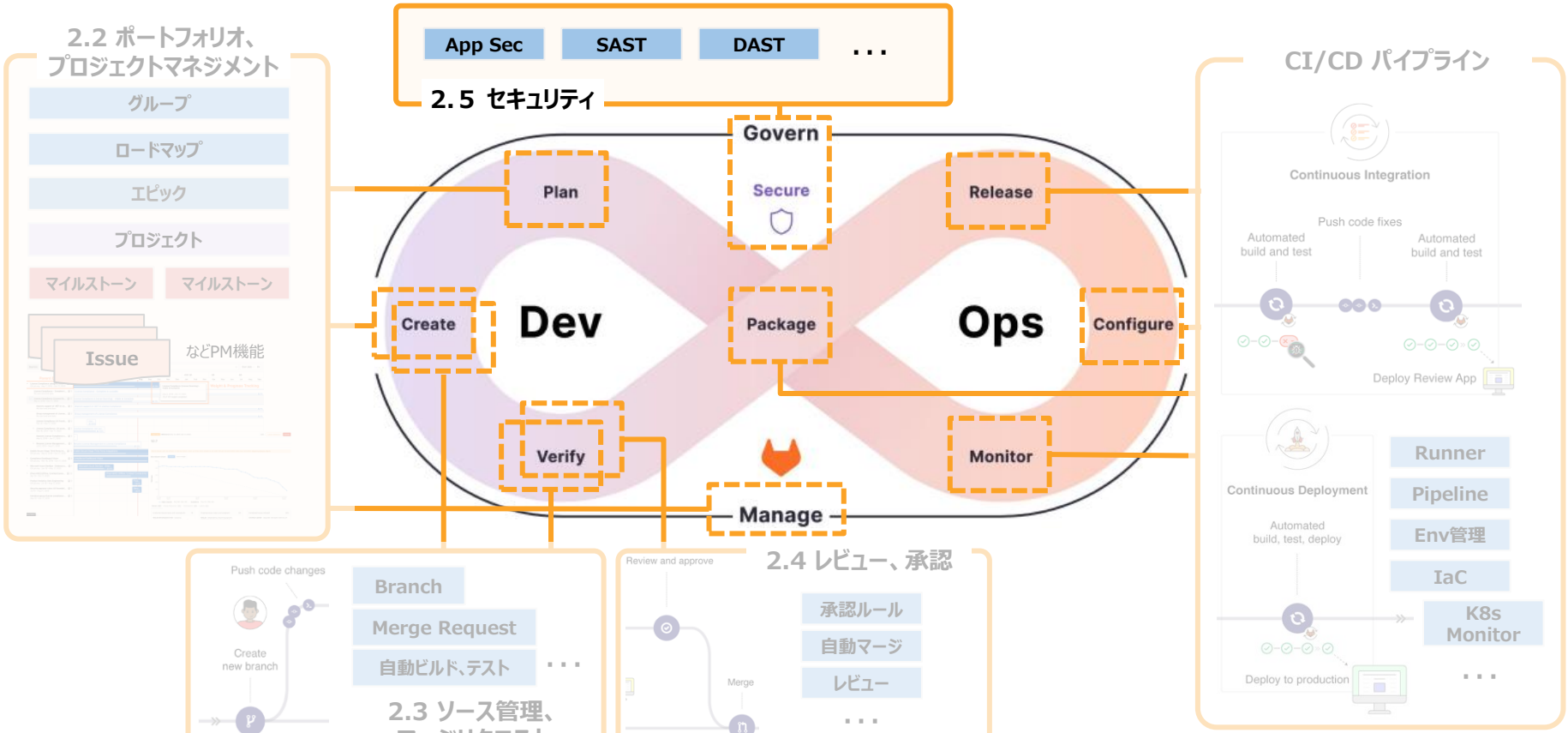
<グループレベル>

- グループ名またはパスの変更
- グループ リポジトリのサイズ制限の変更
- グループの作成または削除
- グループに追加されたユーザー、およびそのアクセス権
- グループ SAMLを介したユーザー サインイン
- グループのユーザーに対する権限の変更
- グループユーザーの削除
- グループにインポートされたリポジトリ
- プロジェクトに共有された情報と、その権限
- プロジェクトに共有されているグループの削除
- LFS の有効化または無効化
- 共有ランナー時間制限の変更
- メンバー追加ロックの有効化または無効化
- リクエスト アクセスを有効化または無効化
- 2FA の実施または猶予期間の変更
- プロジェクトを作成できるロールの変更
- グループ CI/CD 変数の追加、削除、または保護ステータスの変更

<プロジェクトレベル>

- デプロイ キーの追加、削除
- プロジェクトの作成、削除、名前の変更、移動 (転送)、パスの変更
- プロジェクトへのユーザー追加とその権限
- プロジェクトに割り当てられたユーザーの権限変更
- ユーザーの削除
- プロジェクトのエクスポート
- プロジェクトリポジトリのダウンロード
- プロジェクトのアーカイブ
- プロジェクトのアーカイブ解除
- 保護されたブランチの追加、削除、または更新
- リリースの追加
- リリースの更新
- リリース マイルストーンの関連付けの変更
- コミッターによるマージリクエストを承認する権限の更新
- 作成者によるマージリクエストを承認する権限の更新
- 必要な承認の数の更新
- 承認グループへのユーザーとグループを追加または削除
- プロジェクト CI/CD 変数の追加、削除、または保護ステータスの変更
- アクセス トークンの作成または削除
- アクセス トークンの作成または削除の失敗
- デフォルトブランチの変更

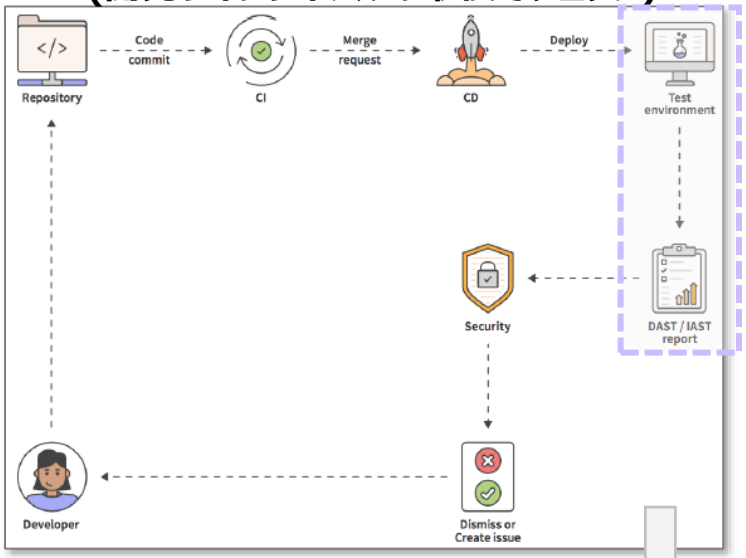
2.5 セキュリティ



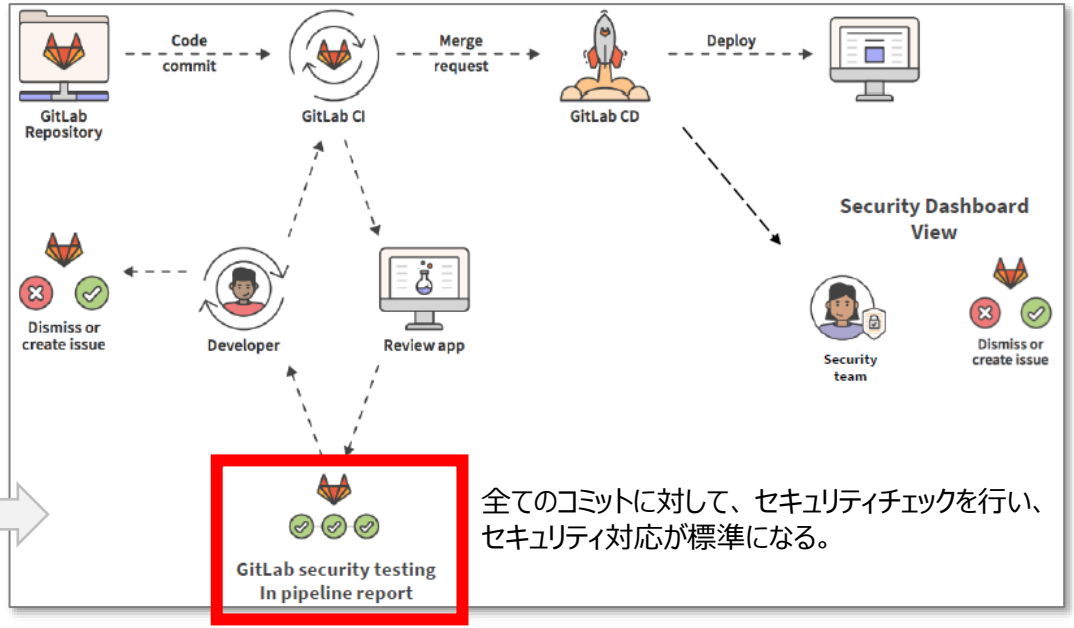
2.5.1 GitLabの継続的アプリケーションセキュリティ

FUJITSU

従来のアプリケーションセキュリティ
(開発ライフサイクルの最後でチェック)

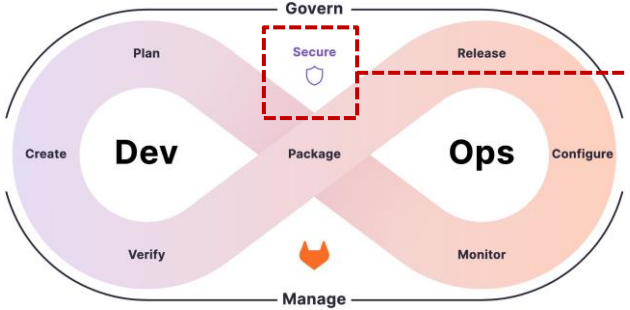


継続的アプリケーションセキュリティ
(Marge前にチェック)

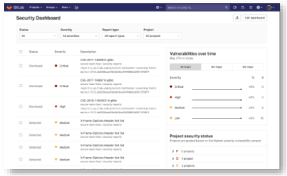


全てのコミットに対して、セキュリティチェックを行い、
セキュリティ対応が標準になる。

2.5.2 アプリケーションセキュリティ一覧



Security Dashboards
プロジェクトの脆弱性を管理するパーソナルスペースで、プロジェクトのデフォルトブランチに存在する脆弱性を可視化



Static Application Security Testing (SAST)
ソースコードを解析することで脆弱性を検出
(例：SQLインジェクション)

Dependency Scanning
パッケージマネージャ（Maven、Gemsなど）の依存関係を解析することで、既知のライブラリの脆弱性を検出

Secret Detection
資格情報、シークレット、パスワードなどの機密情報をチェックし、漏洩を検出

License Compliance
パッケージマネージャの依存関係を解析し、プロジェクトのポリシーに従った承認済みや非許可のライセンスを検出

Container Scanning
Dpkgやrpmなどのパッケージマネージャを解析し、コンテナ内の既知のライブラリの脆弱性を検出

Coverage Guided Fuzz
ホワイトボックス的に関数、メソッドにランダムな引数を渡し、実行

Dynamic Application Security Testing (DAST)
起動中のWebアプリのランタイム脆弱性を検出 （例：CSRF)

Web API Fuzz
予期しない動作やエラーが発生するように、リクエストを発行しテスト

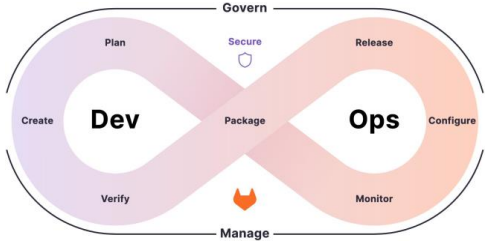
静的スキャン　ランタイムスキャン　動的スキャン

2.6 その他: Value Stream Analytics機能

FUJITSU

Value Stream Analytics

DevOps各ライフサイクルに掛かった時間を集計、可視化することにより、プロジェクトのリリーススピード、プロジェクトの改善Valueを図ります。



GitLab.org › GitLab › Value Stream Analytics

Recent Project Activity

2796	4093	1327	Last 30 days
New Issues	Commits	Deploys	

Stage	Median
Issue	less than a minute
Plan	1 day
Code	3 minutes
Test	about 1 hour
Review	about 22 hours
Staging	9 minutes
Total	4 days

Related Merged Requests	Time
Time between merge request creation and merge/close	
Rescue invalid URLs during badge retrieval in asset proxy !26524 · Opened about 3 hours ago by Lucas Charles	1 hr 30 mins
Ensure more of Sidekiq logs are in JSON !26480 · Opened about 9 hours ago by Stan Hu	2 hrs 21 mins
Resolve "ActionView::Template::Error: undefined method `concat' for nil:Ni... !26395 · Opened 1 day ago by Juan J. Ramirez	14 hrs 37 mins
Fix add EC2 instances step !26377 · Opened 1 day ago by Collen	48 mins
Fix several ID-dependent specs !26372 · Opened 1 day ago by Rémy Coutable	2 hrs 45 mins
Change "Boolean" to lowercase in docs !26361 · Opened 1 day ago by Marcin Sedlak-Jakubowski	13 hrs 21 mins
Wait for the 'Empty wiki' SVG to load before performing test assertions !26309 · Opened 2 days ago by Rémy Coutable	1 day 22 hrs

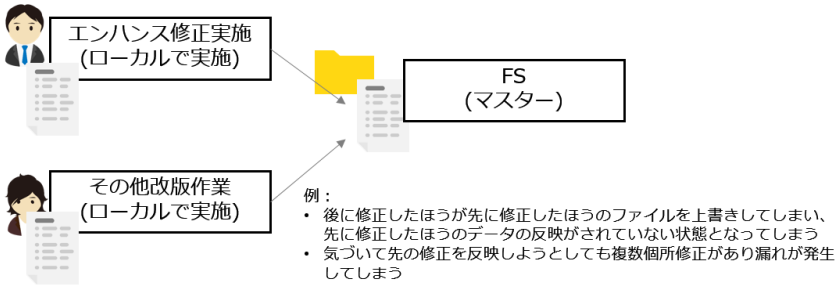
2.7 利用シーン①:協業・共同・委託など開発効率化、明確化

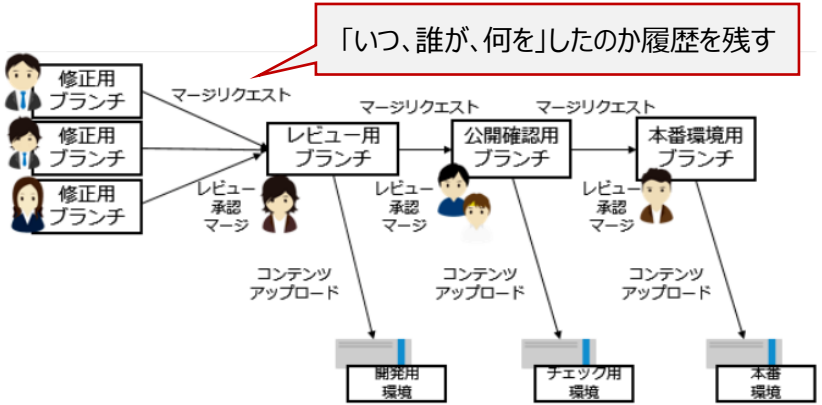
Before

- メールやFTPなどによる資産の受け渡し
- 資産の変更や訂正を目視とコピー&ペーストによるマージに頼る
- メールや会議など記録に残しづらいやりとりが散見
- 権限管理が難しく誤削除などの誤操作を起こす

After

- Gitリポジトリーによる資産の受け渡し
- Merge Requestを活用したレビュー、自動テスト、自動マージ
- Issue、Wikiによるやり取りの記録
- Merge Rule、Protectedブランチによる独自のレビューや承認フローの実現





2.7 利用シーン②:リポジトリ、Wikiによるコミュニケーション改善

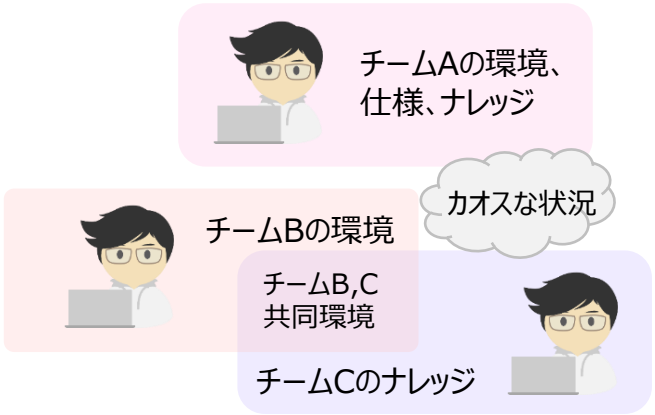
FUJITSU

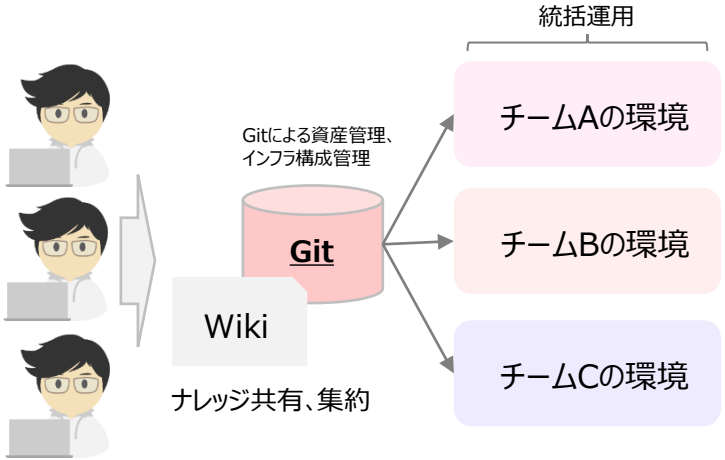
Before

- プロジェクトごとのノウハウが分断して開発ナレッジの共有ができていない
- リポジトリーサーバーが乱立してプロジェクトごとに設計書やソースコードのバージョン管理が混乱
- サーバーの乱立でインフラ運用コストが肥大化

After

- プロジェクトごとのノウハウをWikiで共有して開発ナレッジの集約・共有
- リポジトリーサーバーを集約してGitによるプロジェクトごとの設計書やソースコードのバージョンを管理
- サーバー集約によりインフラ運用コストを削減





2.7 利用シーン③:自社Pipeline、IaCなどの実現

FUJITSU

Before

- プロジェクトごとのノウハウが分断して開発ナレッジの共有ができていない
- リポジトリーサーバーが乱立してプロジェクトごとに設計書やソースコードのバージョン管理が混乱
- サーバーの乱立でインフラ運用コストが肥大化

After

- プロジェクトごとのノウハウをWikiで共有して開発ナレッジの集約・共有
- リポジトリーサーバーを集約してGitによるプロジェクトごとの設計書やソースコードのバージョンを管理
- サーバー集約によりインフラ運用コストを削減

保守T

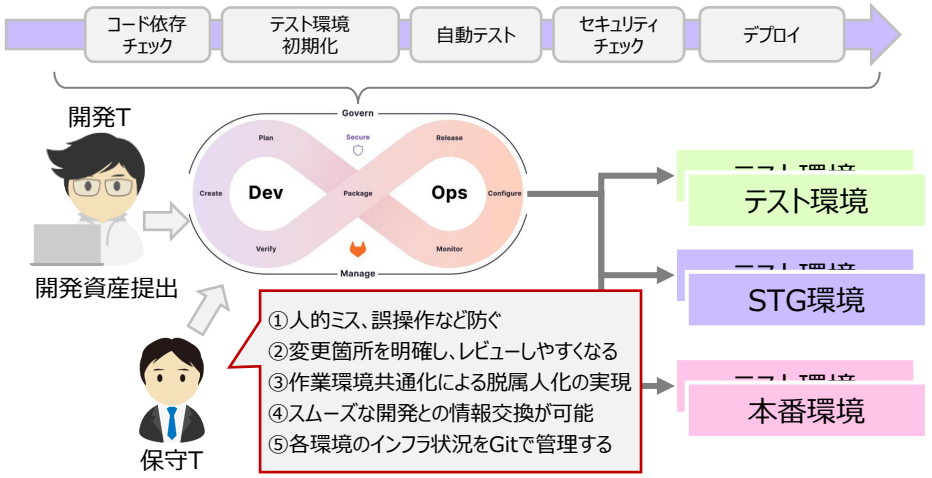
①手作業でデプロイメント、リリースなど対応

②目視でレビュー

③〇〇作業は〇〇さんの環境に依存

④運用環境の設定が異なるため、開発チーム側の設定漏れがよく発生

⑤環境ごとにパラメータが違うため、管理負荷が発生



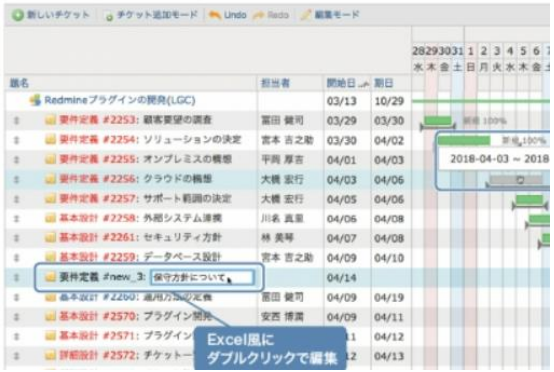
2.7 利用シーン④:組織を超える課題管理、脱Excel活動

Before

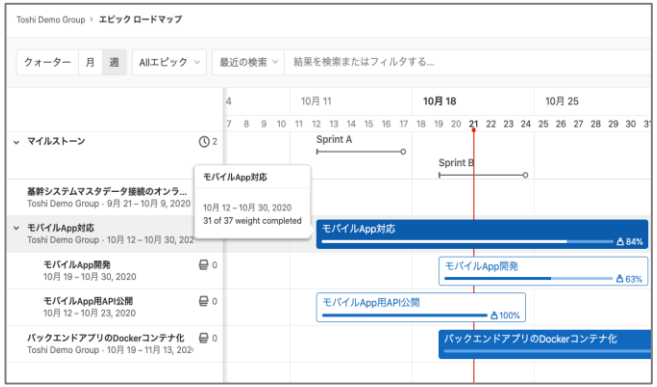
- Excelシートを利用して課題管理を行うために課題の共有・更新が難しい
- 組織を越えたタスク管理が難しい

After

- IssueとBoard 機能により視覚的に課題を管理
- Epic 、Roadmapの機能でプロジェクトを越えた大きな課題を視覚的に管理



Excelでタスク、スケジュール管理



・組織を超えた、各部門、チームレベルのロードマップ管理が簡単になる
・Excelの編集作業が不要で、プロマネに集中できる

2.7 利用シーン⑤:プロジェクトの費用対効果(ROI)を計測

FUJITSU

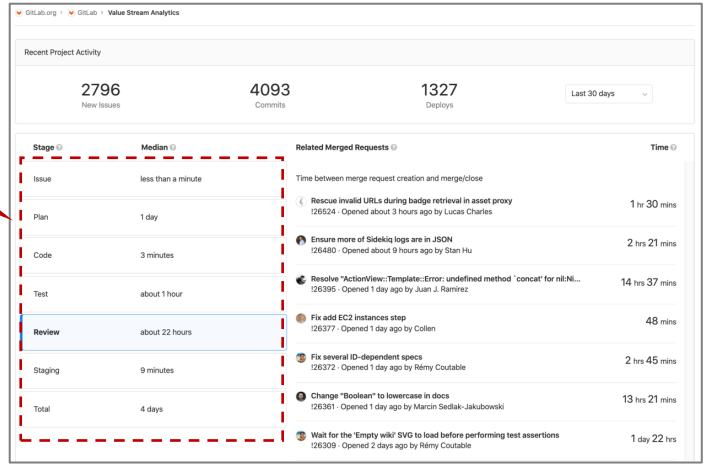
Before

- 各ステップでの対応時間の集計が難しい
- 全体の課題解決時間やバグ改修コストを把握しづらい
- 各メンバーの活動頻度を把握し難い

After

- Value StreamによりIssueの解決時間やPipelineの実行時間を自動的に集計
- Burndown/Burnup chart、Value Streamによるプロジェクトの状態、課題解決状況、組織のROIを計測してActivityでユーザーの活動を見える化

どのStepにどのくらいの時間が掛かったか



① ボトルネックとなった箇所や最も時間を要する作業が特定できる
② ROIを測定するために根拠を数字で示せる
③ DXの取り組みやDevOpsの導入などの効果が明確になる

FUJITSU

3. Digital Application Platformが提供するGitLab製品

3.1 Digital Application Platformが提供するGitLab製品

FUJITSU

1. DevOps with GitLab

- マネージド GitLab を FJcloud-V 上で提供します
- GitLab環境をオンデマンドで提供することで、DevOpsの導入・運用が容易となります

2. GitLab Enterprise Edition Subscription

- DevOps with GitLab を機能拡張するためのライセンス（Premium、Ultimate）をサブスクリプション型で提供します
- お客様の環境（FJcloud-V/Digital Application Platform(※)に限る）にインストールし、セルフマネージド型での利用も可能です
- 30日間Free Trialを提供します

※FJcloud-V/Digital Application Platformには、IaaS/PaaS/DevOps/専有コンポーネント/プライベートリージョン/ゾーンが含まれます

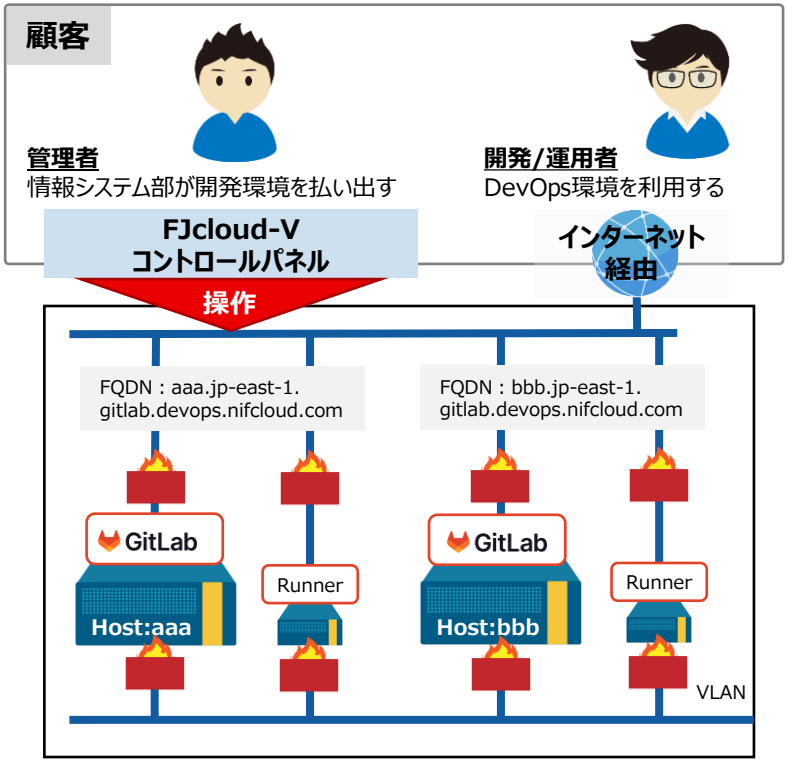
FUJITSU

3.2 DevOps with GitLab

3.2.1 DevOps with GitLabとは

マネージドGitLabをFJcloud-V 上に払い出し、プライベートなDevOps環境を簡単に利用できるサービスです。

- 特長：
  - **The One DevOps Platform**
    - DevOpsに求められる多くの機能を単一の環境で利用可能
  - **プライベートな環境での利用**
    - FJcloud-Vのプライベートな環境でDevOps環境が利用可能
    - GitLab社が提供するSaaSがセキュリティ要件を満たさない場合でもセキュリティを強化したDevOps with GitLabは対応可能
  - **導入・運用が簡単、DevOpsに注力できる**
    - コントロールパネルからGitLab環境を払い出し、DevOpsをすぐに始めることができます
    - インフラ基盤からGitLab環境までマネージドで提供されるためアプリケーションの開発と運用に専念できます



3.2.2 DevOps with GitLab と GitLab社SaaSの違い

FUJITSU

自社要件に合わせて、カスタマイズ可能

専用インスタンスで、セキュリティを強化

		DevOps with GitLab	GitLab.com （GitLab社提供SaaS）
GitLab 一般機能	DevOps ライフサイクル	○	○
	CI実行	専用Hosted Runner提供 (利用無制限)	Shared Runner提供 （時間制限あり）
GitLab 管理機能	GitLab.rbカスタマイズ	○	×
	Admin Area利用	○	×
	独自ドメイン設定	○	×
ネットワーク 設定	プライベートLAN	○	×
	独自ファイアウォール設定	○	×
システム構成	GitLabインスタンス	専有インスタンス	共有インスタンス (他のお客様も同一GitLabを利用)
インフラ運用	サーバー運用	不要	不要
	バージョンアップ 対応	不要	不要
サポート	サポート窓口	富士通サポート（日本語）	GitLabサポート（英文）

凡例 ○:正式提供、△:提供予定、×:未提供

3.2.3 オンプレ構築とマネージドサービスの比較

FUJITSU

○ マネージドサービスを活用し、運用負荷を軽減することで本業へ集中することが可能

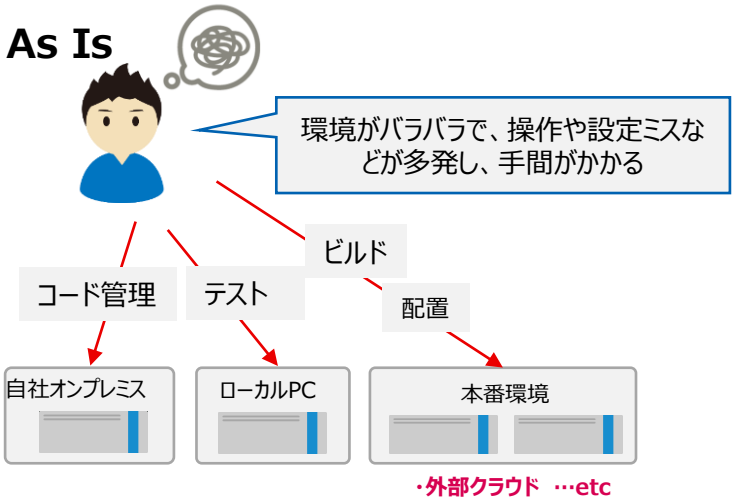
		オンプレで構築/運用	DevOps with GitLab
GitLab管理機能	設定のカスタマイズ	**可能**	一部制限有り
ネットワーク設定	接続NW	グローバルNWの他に **自社NWへの接続も可**	グローバルNW上で提供、 L2通信でプライベートLANへの接続も可
システム構成	HA構成	お客様で対応	**サービス提供**
インフラ運用	バックアップ	お客様で対応	**サービス提供**
	Runnerサーバー	お客様で対応	サービス提供予定(※開発中)
	セキュリティ対策	お客様で対応	**通信ポート制限/脆弱性スキャン等の対策 をFJcloud-V基準で実施**
	ログ管理	お客様で対応	**FWの通信ログ提供** (通信ログ以外はGitLabのログ確認可能)
	バージョン管理	お客様で対応	**サービス提供**(公式バージョンに準ずる、 脆弱性パッチ提供も実施)
その他	コスト	ー	・スモールスタート可(利用後のサーバタイプ変更可) ・基盤運用まで含めるとオンプレと比較して安価になるケースあり

運用負荷軽減

3.2.4 DevOps with GitLab による開発スタイルの変化

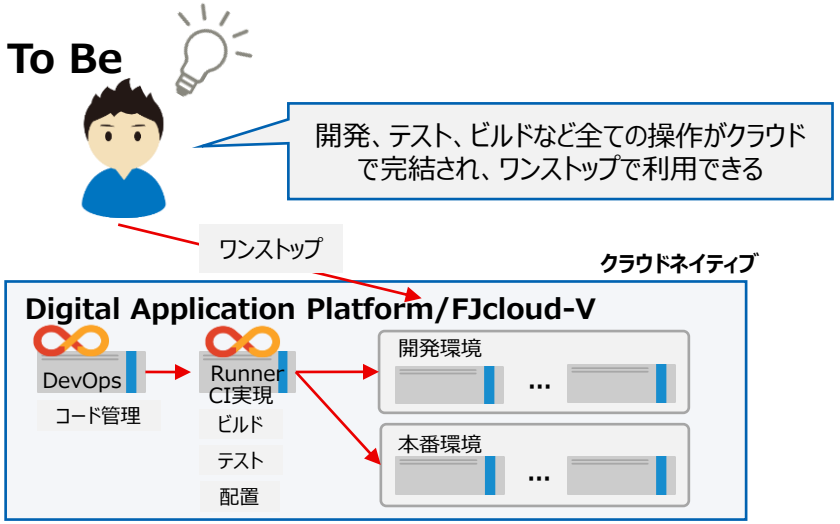
従来の姿

1）自社のオンプレミスのSVNでコードや成果物を管理します。
2）普段は個人のローカルPCで開発・テストを行い、成果物をSVNに提出します。
3）本番環境に適用する場合は、手動でビルドを実行して、SSHを使用して外部のクラウド環境に配置します。



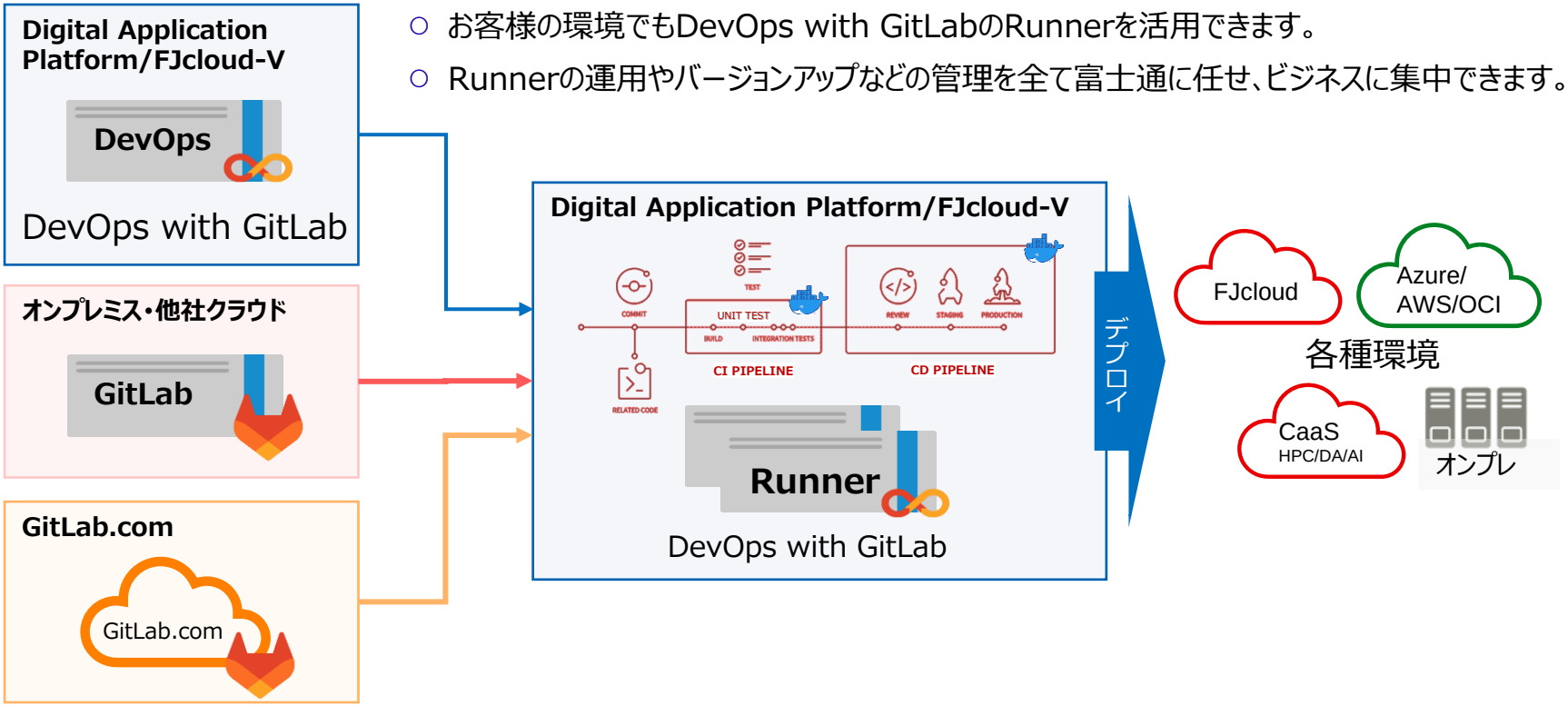
DevOps with GitLab導入後

1）GitLabを使用したDevOpsにより、コード管理とプロジェクト管理を行います。
2）すべての開発とテストはクラウド環境で行い、成果物をGitLabに提出する
3）本番環境と開発環境のビルド、配置、および環境設定は、Runnerを使用して自動化され、手動での作業は必要ありません。



3.2.5 DevOps with GitLab の Runner (CI/CD)機能

- 専用のCI/CD Pipeline環境を簡単に立ち上げ、DevSecOpsサイクルを実行できます。
- お客様の環境でもDevOps with GitLabのRunnerを活用できます。
- Runnerの運用やバージョンアップなどの管理を全て富士通に任せ、ビジネスに集中できます。



3.2.6 DevOps with GitLab 技術仕様・提供価格など

FUJITSU

- DevOps with GitLab 仕様・機能
  https://pfs.nifcloud.com/service/devops.htm

- 提供価格（税込）

  https://pfs.nifcloud.com/price/#devops_with_gitlab

- ドキュメント
  - ユーザーガイド：https://pfs.nifcloud.com/guide/devops/
  - コントロールパネルヘルプ：https://pfs.nifcloud.com/help/devops/

- API

  https://pfs.nifcloud.com/help/devops/

FUJITSU

3.3 GitLab EE Subscription

3.3.1 GitLab EE Subscription

FUJITSU

- 「DevOps with GitLab」と組み合わせて利用可能なGitLab EE（Enterprise Edition）のライセンスをサブスクリプション型で提供します
- 本サービスは2種類（Premium/Ultimate）のサブスクリプションプランを提供し、Free版の機能に加え、企業内でのプロジェクト管理やセキュリティ・コンプライアンス管理を強化することができます。

Free （DevOps with GitLabにて提供）	Premium	Ultimate
アプリケーションのビルド・デプロイ・実行をサポートします。	インクリメンタルなデプロイや先進的なKubernetes管理などの高度なサポートによって、ITの進化を加速させます。	優先度、セキュリティ、リスク、コンプライアンスを管理すると同時に、デリバリを最適化し、ITでのビジネスの進化を加速させます。
主な機能 ● コミュニティサポート ● 統合されたCI/CD ● 課題ボード ● 任意のマージリクエスト承認 ● Wikiによるプロジェクトのドキュメント ● プロジェクト管理のバリューストリーム分析 ● Mattermostとの連携 ● コンテナの脆弱性スキャン ※	**Free に加えて…** ● 複数グループの課題ボード ● エピック / ロードマップ / マージリクエストのルール設定 ● プッシュルール / リポジトリプルミラーリング ● 災害復旧機能 / 監査イベント ● LDAP/ADサーバーのサポート ● 複数プロジェクトのパイプライングラフ ● 高可用性のアーキテクチャをサポート ● GitLab Geoでグローバルな分散型チームをサポート ● パフォーマンス・バーンダウンチャート ● パフォーマンステスト / 環境ダッシュボード	**Premium に加えて…** ● 依存関係スキャン ● 動的アプリケーションセキュリティテスト ● 機密情報の検出 / 脆弱性管理 ● セキュリティダッシュボード / インサイト ● セキュリティダッシュボード ● コンテナネットワークポリシーの管理 ● コンテナの脆弱性スキャン ● コンプライアンスダッシュボード ● クレデンシャルイベント ● 障害発生時の自動ロールバック
ー	複数プロジェクトの管理が必要な組織向け	複数プロジェクト管理に加え、セキュリティやコンプライアンス管理も重視する組織向け

※最新版:コンテナの脆弱性スキャンは無償で利用可能。DevOps with GitLabにて提供中

3.3.2 サービス仕様・提供価格などについて

FUJITSU

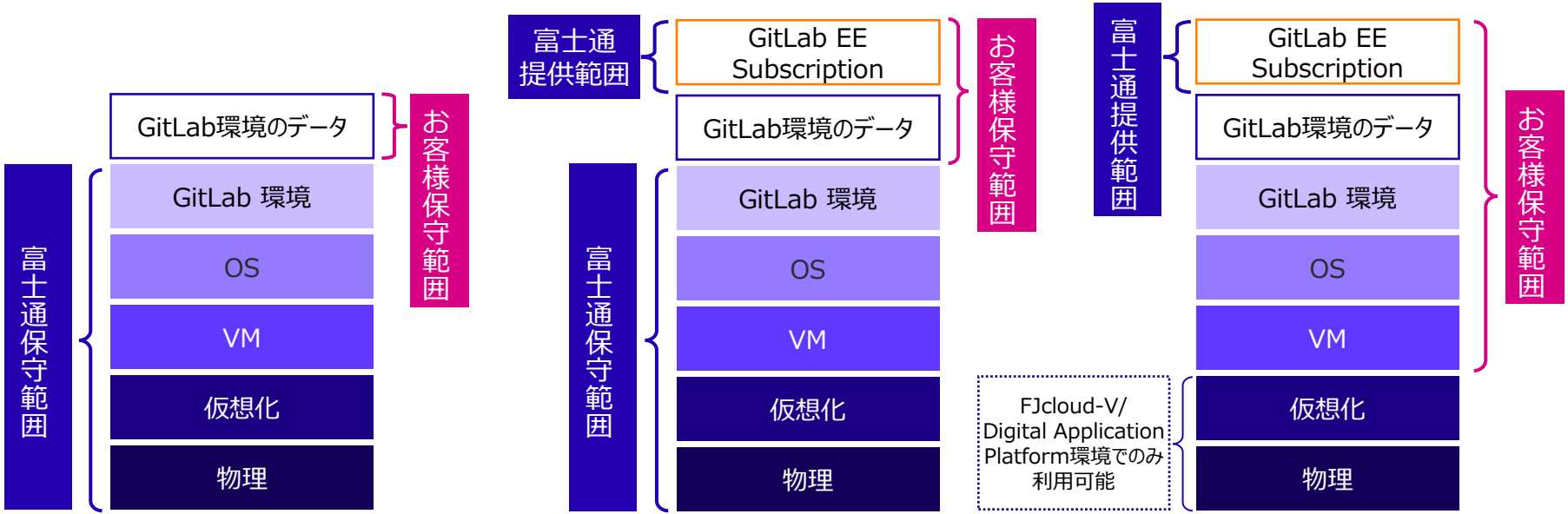
- GitLab Enterprise Edition サブスクリプション 仕様・利用方法・申込み
  https://pfs.nifcloud.com/service/gitlab_ee_sub.htm

- 提供価格（税込）
  https://pfs.nifcloud.com/price/#gitlab

- 30日間Free Trial申込み
  https://inquiry.nifcloud.com/webeq/pub/cloud/gitlabsub_trial_auth

3.4 ご利用形態ごとの責任分担

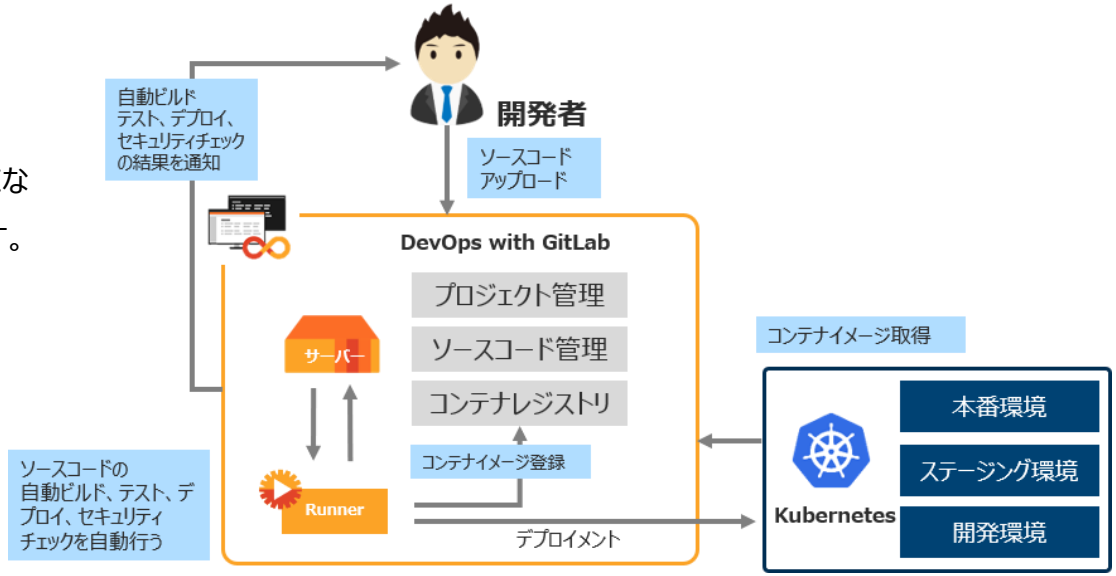
○「DevOps with GitLab」、「DevOps with GitLab + GitLab EE Subscription」、「GitLab EE Subscription」のそれぞれのご利用形態ごとの責任分担は以下の通りです。



3.5 構成例（1）：CI/CDを構築したい

Kubernetes基盤をモニターしたい、自前のCI/CDを構築したい

- CI/CD環境の構築やDevOps環境の運用において、簡単に実行できるようになり、サーバーの管理が不要になることで、ユーザーの運用負荷を軽減することができます。

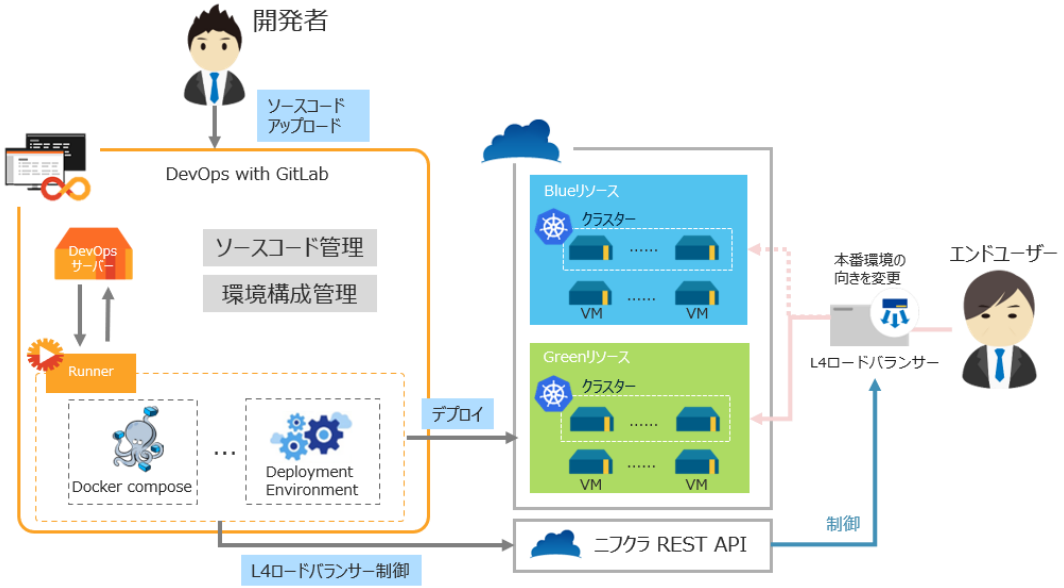


・CI/CD パターン: https://pfs.nifcloud.com/cdp/other/ci_cd.htm

3.5 構成例（2）:Blue/Greenデプロイメントを実現したい

Blue/Greenデプロイメントを実現したい

- GitLabとL4ロードバランサーを利用すれば、Blue/Greenデプロイメントが実現可能。
- インフラリソースを並行して利用し、問題があれば簡単に旧バージョンに戻せる。
- 運用負荷を軽減し、開発に集中できる。
- L4ロードバランサーを使って、バージョンアップ時のサービス停止時間を最小限にできる。
- 旧環境を開発環境として利用することで、出費を抑えられる。

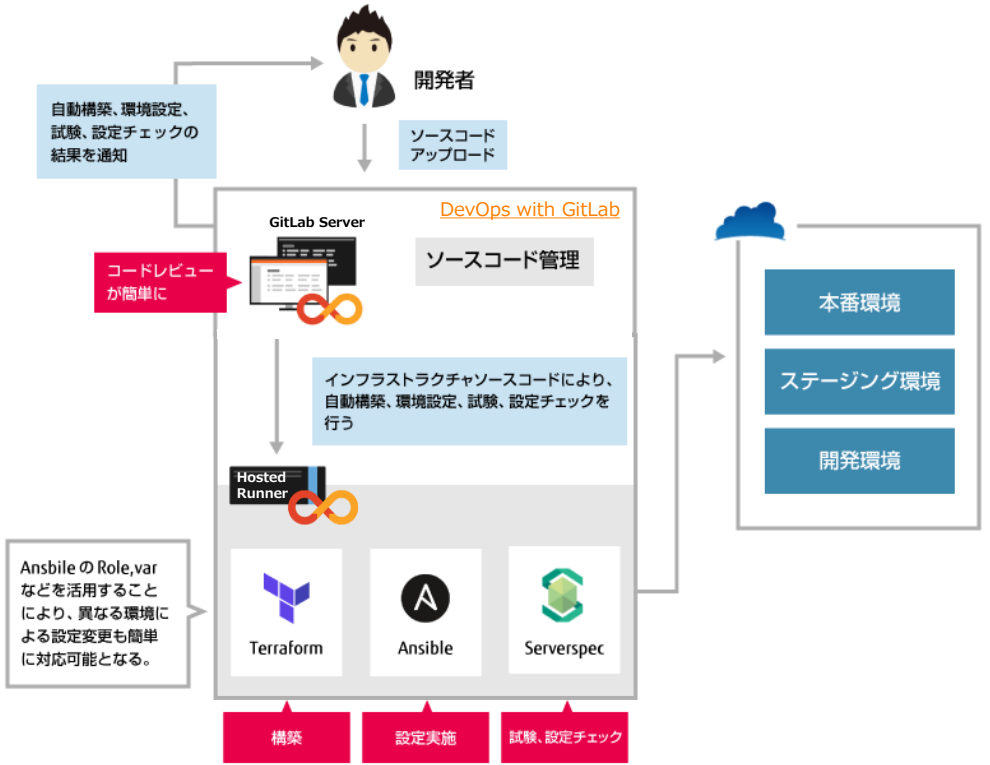


・Blue/Green デプロイメントパターン:https://pfs.nifcloud.com/cdp/other/blue_green.htm

3.5 構成例（3）:Infrastructure as Code(IaC)の実現

Infrastructure as Code（IaC）の実現

- ソースコードを使用してインフラストラクチャを構築し、ニフクラ DevOps with GitLabで管理することで、メンテナンス性を向上できます。
- 変更箇所が明確になり、レビュー作業も効率的に進められます。
- CI/CDサーバーによる自動化により、ミスや属人化を減らし、環境構築作業のコストと時間を削減することができます。



・Infrastructure as Code（IaC）パターン:https://pfs.nifcloud.com/cdp/other/iac.htm

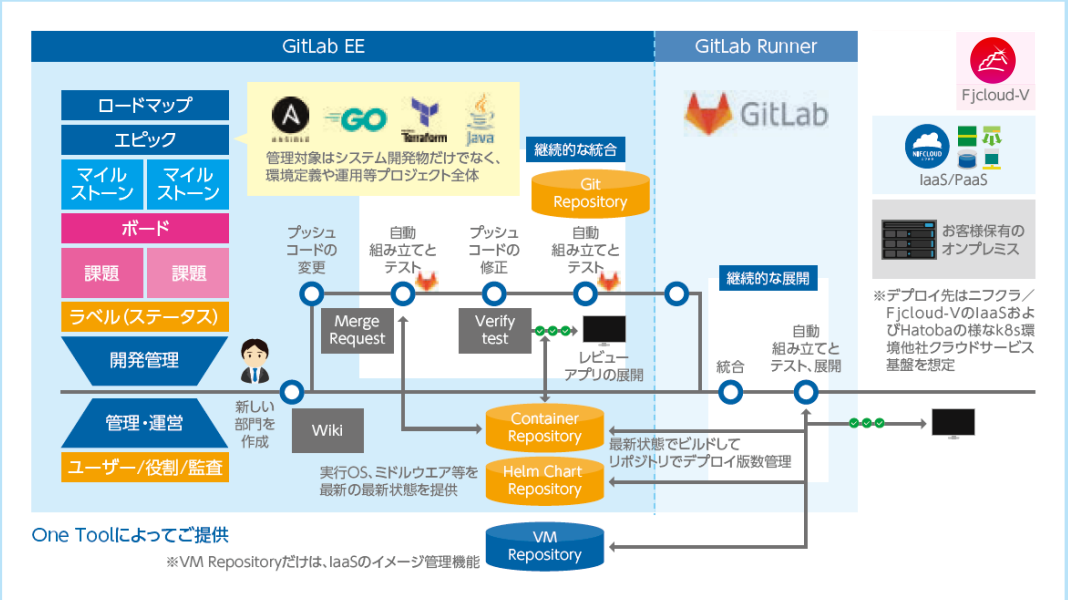
3.6 導入事例（1）：グループ会社のDX基盤として導入

- 富士通グループのクラウド開発会社である富士通クラウドテクノロジーズは、全社共通のDevOps基盤として、GitLab EE Premiumを採用しました。
- 優れた権限管理を活用して、オープンソースソフトウェア（OSS）文化に基づくDevOpsを実現しています。
- その結果、生産性が向上し、リリース期間が短縮され、部門間でのノウハウ共有が可能になりました。
- 利用層は非エンジニアにも広がり、課題管理や監査対応にも役立っています。

事例詳細:
https://pfs.nifcloud.com/cs/catalog/cloud_caseinterview/catalog_20210806100000_1.htm

課題の解決

- ノウハウの共有で全社的に生産性とリリース頻度が向上



3.6 導入事例（2）:ソース管理ツール刷新、分散開発実現

○ GitLabマネージドサービスの導入により、ソース管理を効率化

顧客課題

- 老朽化した既存ソース管理ツールの移行とともに、グループ内情報子会社とのオフショア分散開発に対応したい
- 自社のみでのGitLabの導入や運用は担当者への負荷とコスト面が懸念
- クラウド上でのGitLab導入はセキュリティ面に不安がある

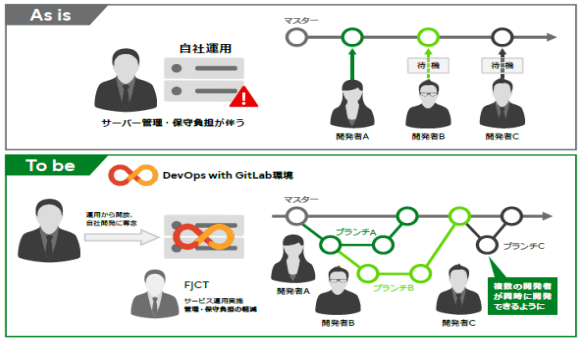
解決策

- 必要な機能が網羅されたDevOps with GitLabを選択
- マネージドサービスでの提供で、インフラ構築・運用の負荷を大幅に抑制
- FJcloud-Vの多彩なセキュリティ機能により安心安全なクラウド環境を構築

Why Fujitsu

- **必要な機能がすべて備わったGitLab環境を簡単に構築・利用**できるため、アプリケーションの開発と運用に専念可能に
- **セキュリティ面でも堅牢なFJcloud-V上のGitLab環境**を利用することで、自社によるインフラの構築・運用が不要
- 基礎教育の実施から移行／構築支援、定例会の開催など、手厚いサポートサービスの提供により、スムーズな導入と**GitLabに関する知見・ノウハウを取得**

導入前後イメージ



https://jp.fujitsu.com/solutions/cloud/fjcloud/-v/casestudies/ess/

Thank you