

CREATIONLINE, Inc
Cloud Solution Division
DevOps Solution Team

Yosuke Imai

GitLabで繋ぐ、原点から未来へ

DevSecOpsプラットフォームだからこそできる

GitLabのValue Stream分析とAI活用

● 01 DevSecOpsプラットフォームなGitLab

● 02 GitLab Value Stream Analytics

● 03 GitLab Duo

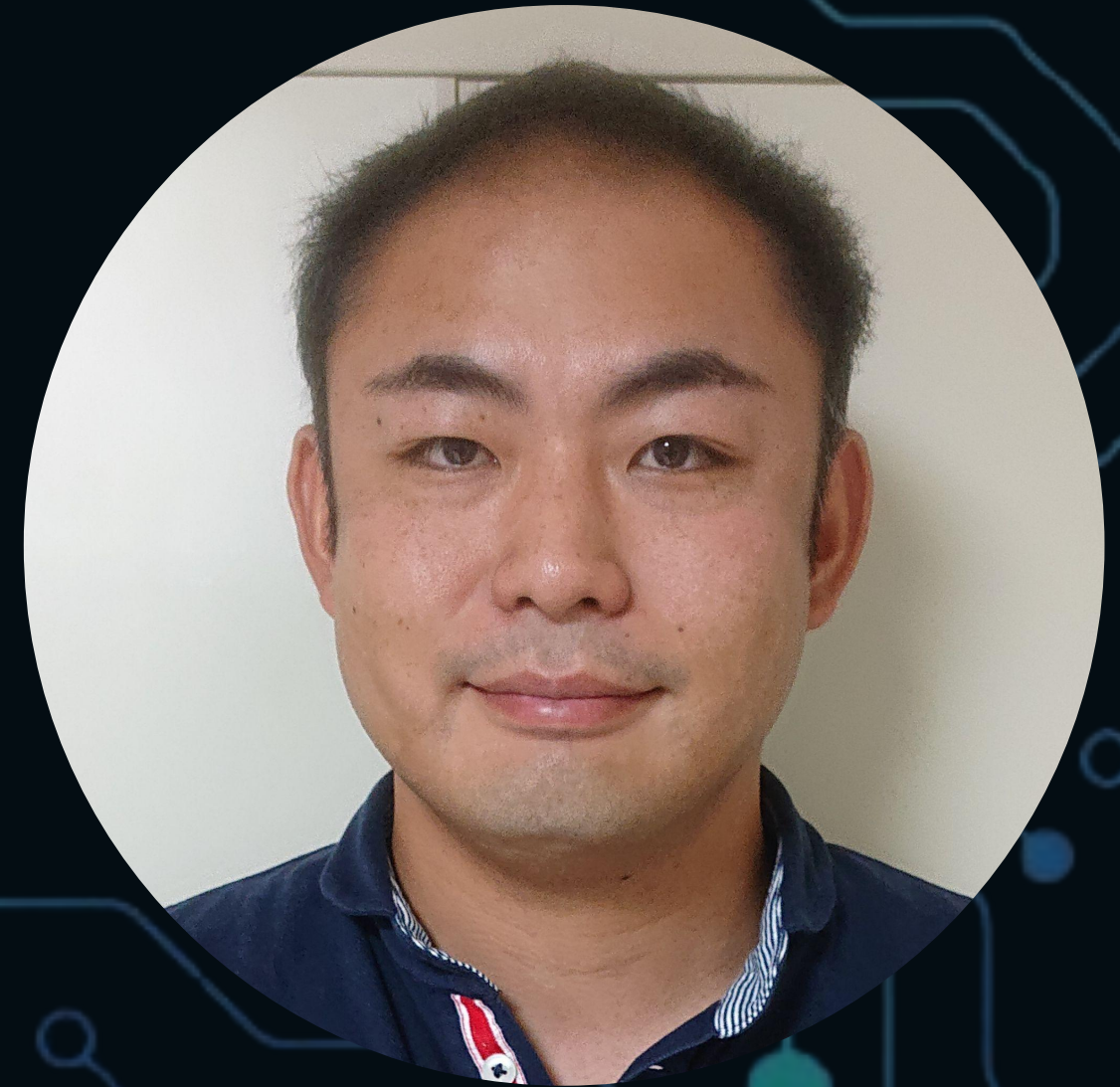
CONTENTS

自己紹介

今井 陽祐

YOSUKE IMAI

CREATIONLINE, Inc
Cloud Solution Division
DevOps Solution Team
GitLab Team Leader
Customer Success Manager

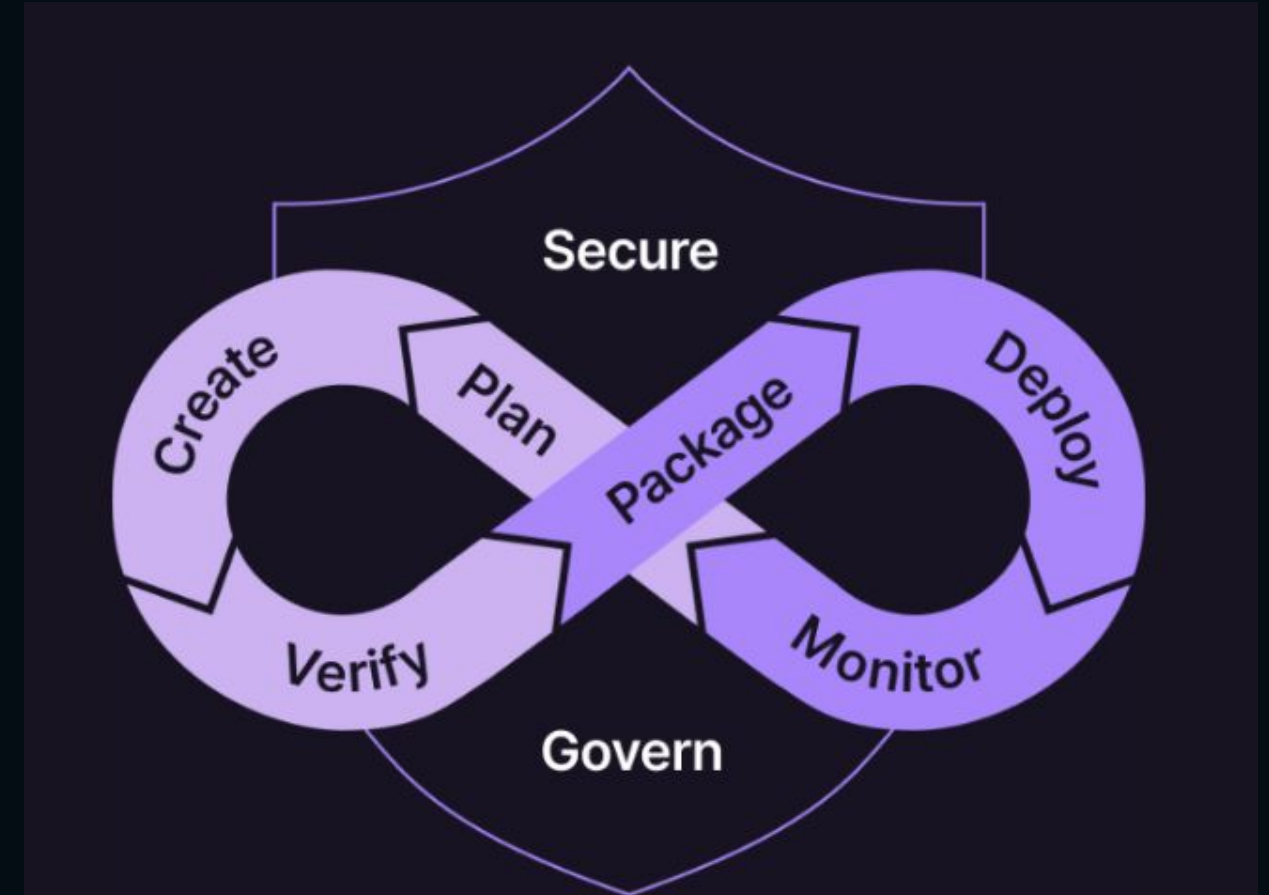


Section
01

DevSecOpsプラットフォームなGitLab

01 DevSecOpsプラットフォームなGitLab

Value Stream全体をカバーしているGitLab



Plan	Create	Verify	Secure	Package	Deploy	Monitor	Govern
Pages	Code	Review Apps	Container	Dependency	Infrastructure	Service Desk	Software Bill of
Wiki	Suggestions	Merge Trains	Scanning	Proxy	as Code	Incident	Materials
Value Stream	GitLab CLI	Code Testing	Software	Helm Chart	Deployment	Management	Dependency
Management	Web IDE	and Coverage	Composition	Registry	Management	On-call Schedule	Management
Portfolio	Code Review	Continuous	Analysis	Container	Auto DevOps	Management	Vulnerability
Management	Workflow	Integration (CI)	Fuzz Testing	Registry	Environment		Management
Team Planning	Source Code	Secrets	API Security	Package	Management		Compliance
DORA Metrics	Management	Management	DAST	Registry	Release		Management
DevOps Reports	Remote	CI/CD Visibility	Secret Detection	Dependency	Orchestration		Audit Events
	Development		SAST	Firewall	Feature Flags		Security Policy
			Code Quality		Continuous		Management
					Delivery		Release Evidence

Section
02

GitLab Value Stream Analytics

DevOpsの成功とは？

DevOpsが成功しているかどうかの判断は非常に困難です

売上/利益が上がり、ビジネスが成功していればDevOpsも成功していると言えるのでしょうか？



DevOpsの成功とは？

それだけではなく、変化の激しい時代に適応するため、ユーザーからのフィードバックを素早く取り入れ、短いサイクルでソフトウェアを変更していく能力が求められます

つまり、チームのパフォーマンスが高まっていることも大事です



チームパフォーマンスを示す DORA Four Keys

Four Keysとは、GoogleのDORA (DevOps Research and Assessment) チームが割り出したソフトウェア開発チームのパフォーマンスを示す4つのメトリクスです

このメトリクスを使って、自分たちのチームのパフォーマンスを計測したいと考えたとき、必要な情報を揃えるのは困難です



GitLabのValue Stream Analytics

GitLabのValue Stream Analyticsはソフトウェア開発ライフサイクルをend-to-endで可視化できます

すなわちGitLab使って開発を行えば、自然とDORA Four Keysに必要な情報が揃います

GitLabのDORA Four Keysの測定方法

デプロイの頻度

1日あたりの本番環境へのデプロイ数

変更のリードタイム

コードがコミットされてから本番環境で稼働するまでの日数

サービス復元時間

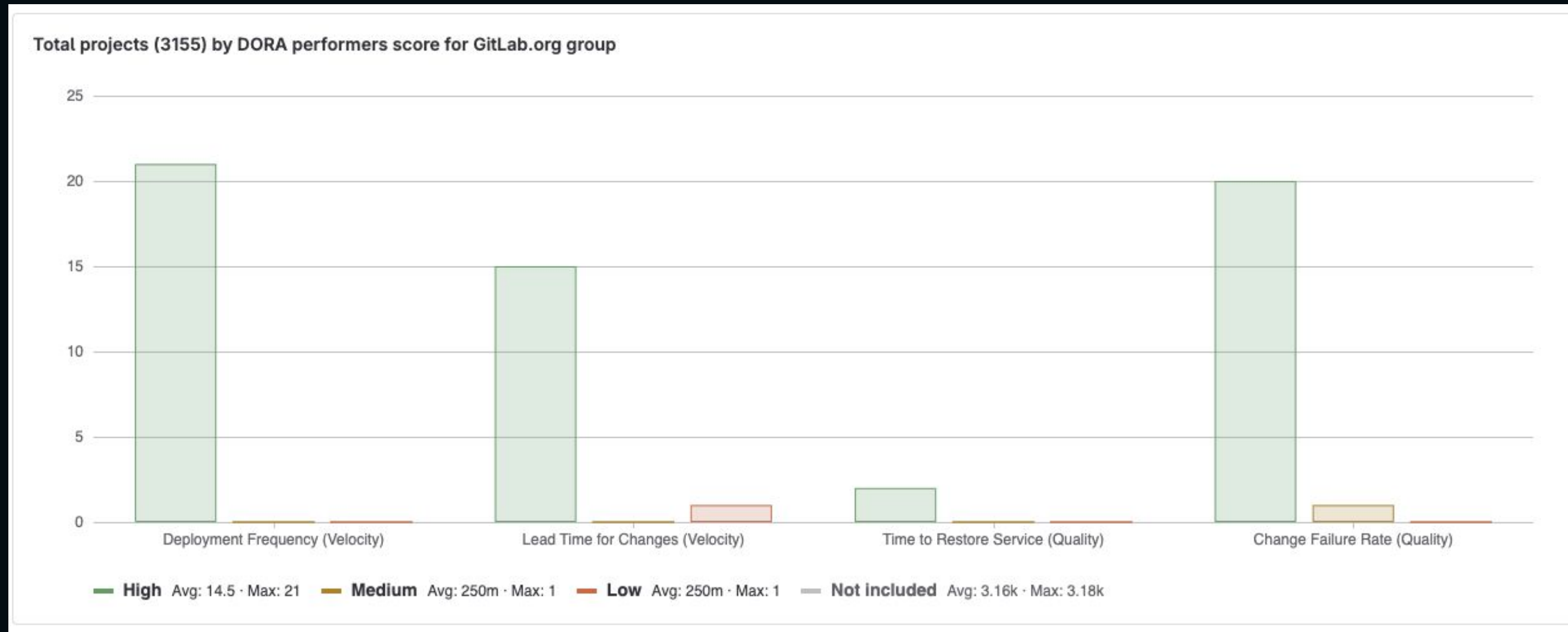
インシデントタイプのインシチュアが作成されてから、本番環境へデプロイするまでの日数

変更障害率

特定の期間におけるインシデントの数と本番環境へのデプロイ数を割ったもの

02 GitLab Value Stream Analytics

DORA Performers score panel (16.2)



https://docs.gitlab.com/ee/user/analytics/value_streams_dashboard.html#dora-performers-score-panel

測定方法の カスタマイズ

先ほど紹介した測定方法はデフォルトの測定方法です

GitLabでは実際の開発フローに合わせる形で、測定方法をカスタマイズすることが可能です

Create value stream

Value Stream name

Custom DORA metrics

Create from default template

Create from no template

Stage 1

サービス復元時間

Start event

Issue label was added

Start event label

LifeCycle::incident

End event

Issue label was added

End event label

LifeCycle::Deployed

Cancel Add another stage Create value stream

02 GitLab Value Stream Analytics

測定方法の カスタマイズ

先ほど紹介した測定方法はデフォルトの測定方法です

GitLabでは実際の開発フローに合わせる形で、測定方法をカスタマイズすることが可能です

The screenshot displays the GitLab Value Stream Analytics interface. At the top, there's a 'Scoped labels' section with a 'Premium' badge and 'All offerings' link. Below this, three labels are listed: 'devops configure', 'devops create', and 'devops defend', each with a description and a 'Subscribe' button. A red arrow points from the 'devops defend' label to the 'End event label' dropdown in the 'Create value stream' configuration panel. The 'Create value stream' panel shows 'Value Stream' set to 'Custom DO', 'Stage 1' set to 'サービス復元', 'Start event' set to 'Issue label was added', and 'End event' set to 'Issue label was added'. The 'End event label' dropdown is highlighted with a red box and contains the selected label 'LifeCycle::Deployed'. The 'Start event label' dropdown also contains 'LifeCycle::incident'. At the bottom, there are buttons for 'Cancel', 'Add another stage', and 'Create value stream'.

測定方法の カスタマイズ

先ほど紹介した測定方法はデフォルトの測定方法です

GitLabでは実際の開発フローに合わせる形で、測定方法をカスタマイズすることが可能です

The screenshot displays the GitLab Value Stream Analytics interface. At the top, there's a 'Scoped labels' section with a 'Premium' badge and 'All offerings' link. Below this, a list of labels is shown, including 'devops configure' and 'devops defend'. A red box highlights the text: 'Labelを使った計測のカスタマイズに非常に便利な機能' (A very convenient feature for customizing measurements using labels). Below this, the 'Create value stream' configuration is visible. It shows 'Value Stream' set to 'Custom DO', 'Stage 1' set to 'サービス復元', and 'Start event' and 'End event' both set to 'Issue label was added'. A red box highlights the 'End event label' dropdown, which is set to 'LifeCycle::Deployed'. A red arrow points from this dropdown to the 'devops defend' label in the list above. At the bottom, there are buttons for 'Cancel', 'Add another stage', and 'Create value stream'.

Section
03

GitLab Duo

AI-powered workflows

Value Stream全体で、AIの力を借り、効率を上げ、サイクルタイムを短縮します



Privacy-first, enterprise-grade

プライバシーを第一に考えたアプローチで、企業や規制対象組織がAIを活用したワークフローを導入できるよう支援します



Single application

セキュリティが組み込まれたシングルアプリケーションにより、より多くのソフトウェアをより速く提供し、バリューチェーン全体で経営陣の可視性を可能にし、コンテキストスイッチを防止します



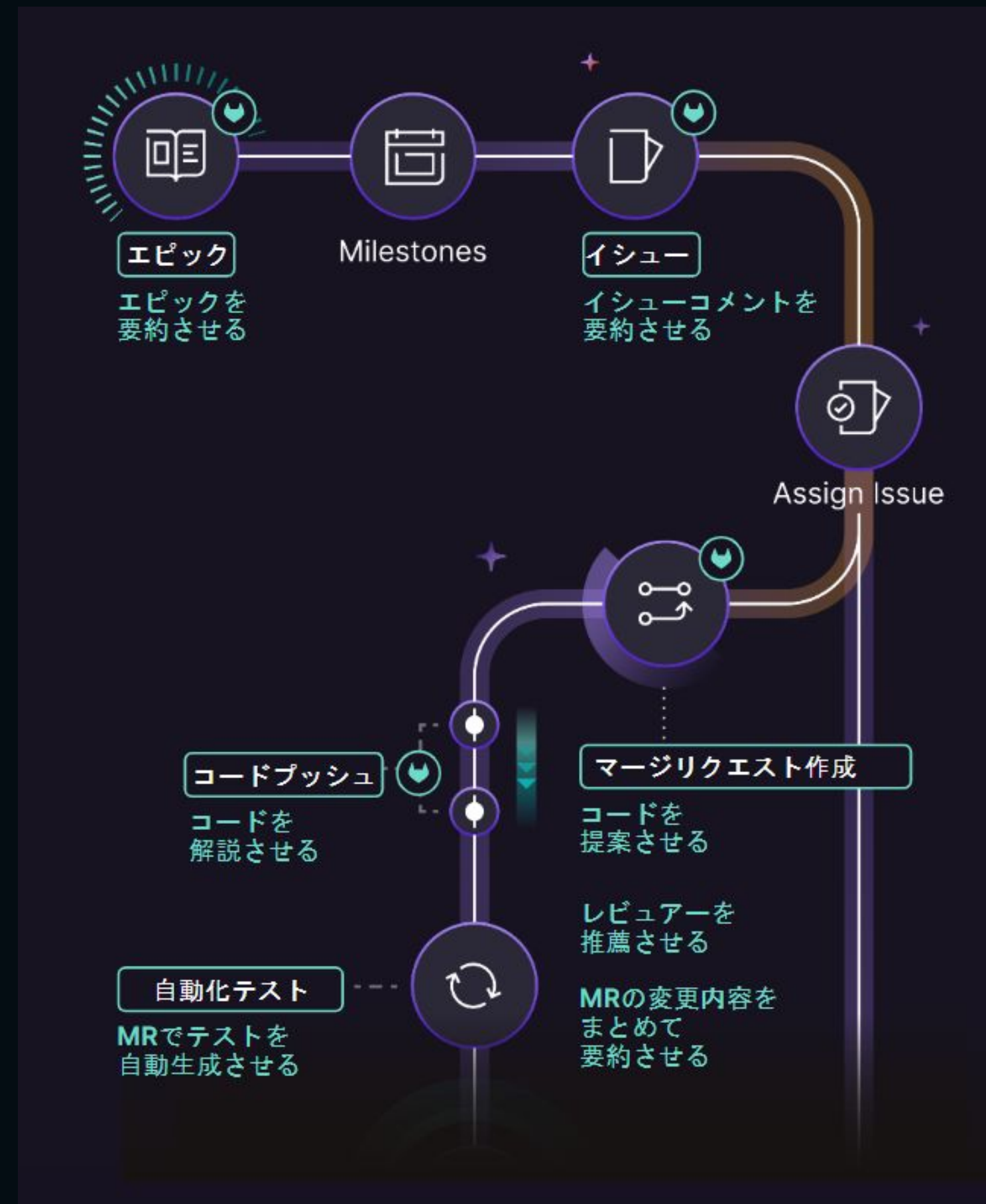
GitLabのAI活用機能の特徴

コードを書いてもらうだけでなく、Value Stream
のあらゆる場面でAIに手助けしてもらう



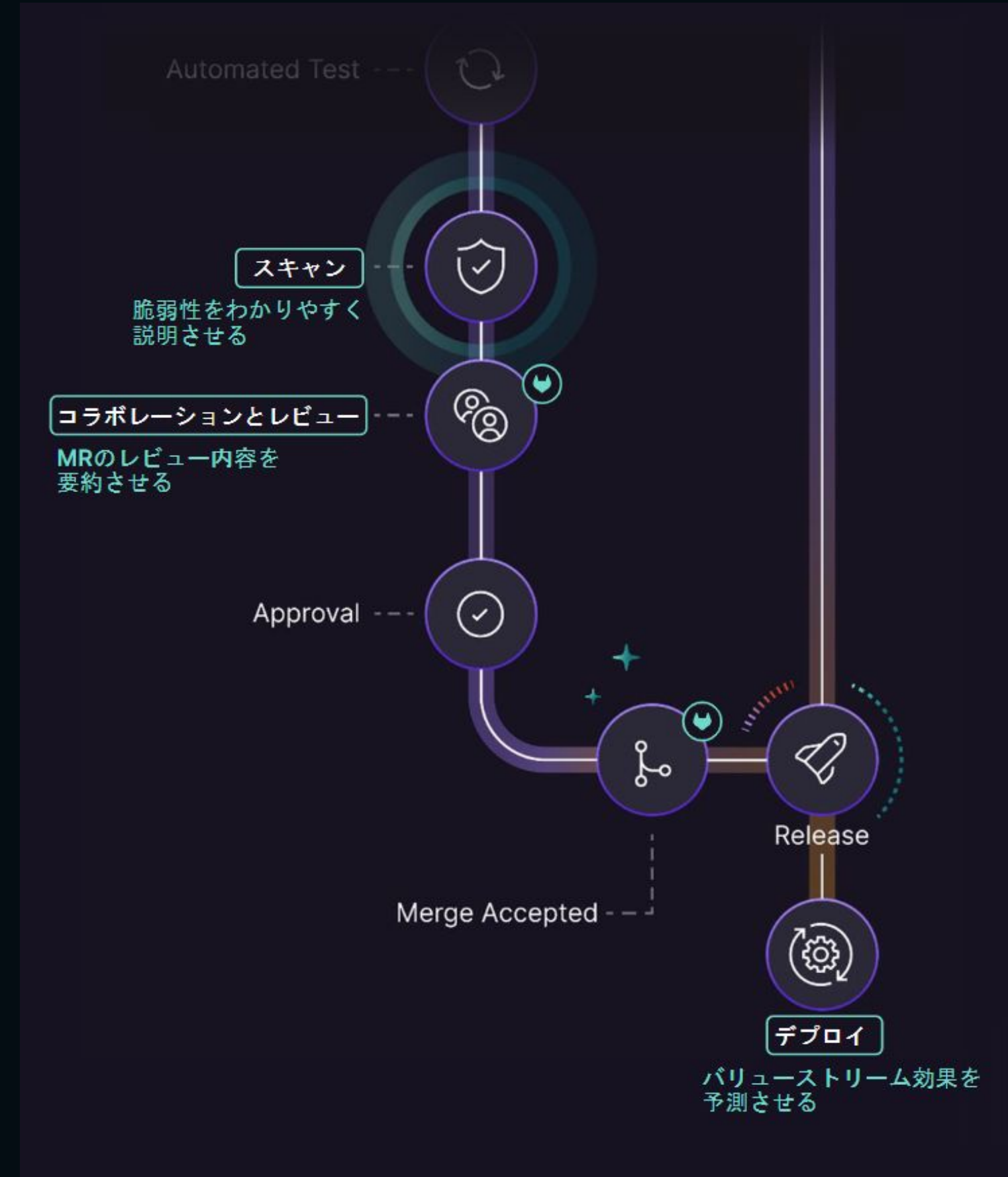
GitLabのAI機能の登場シーン

GitLabはDevSecOpsプラットフォームの名の通り、Value Streamの随所でAI機能が登場します。



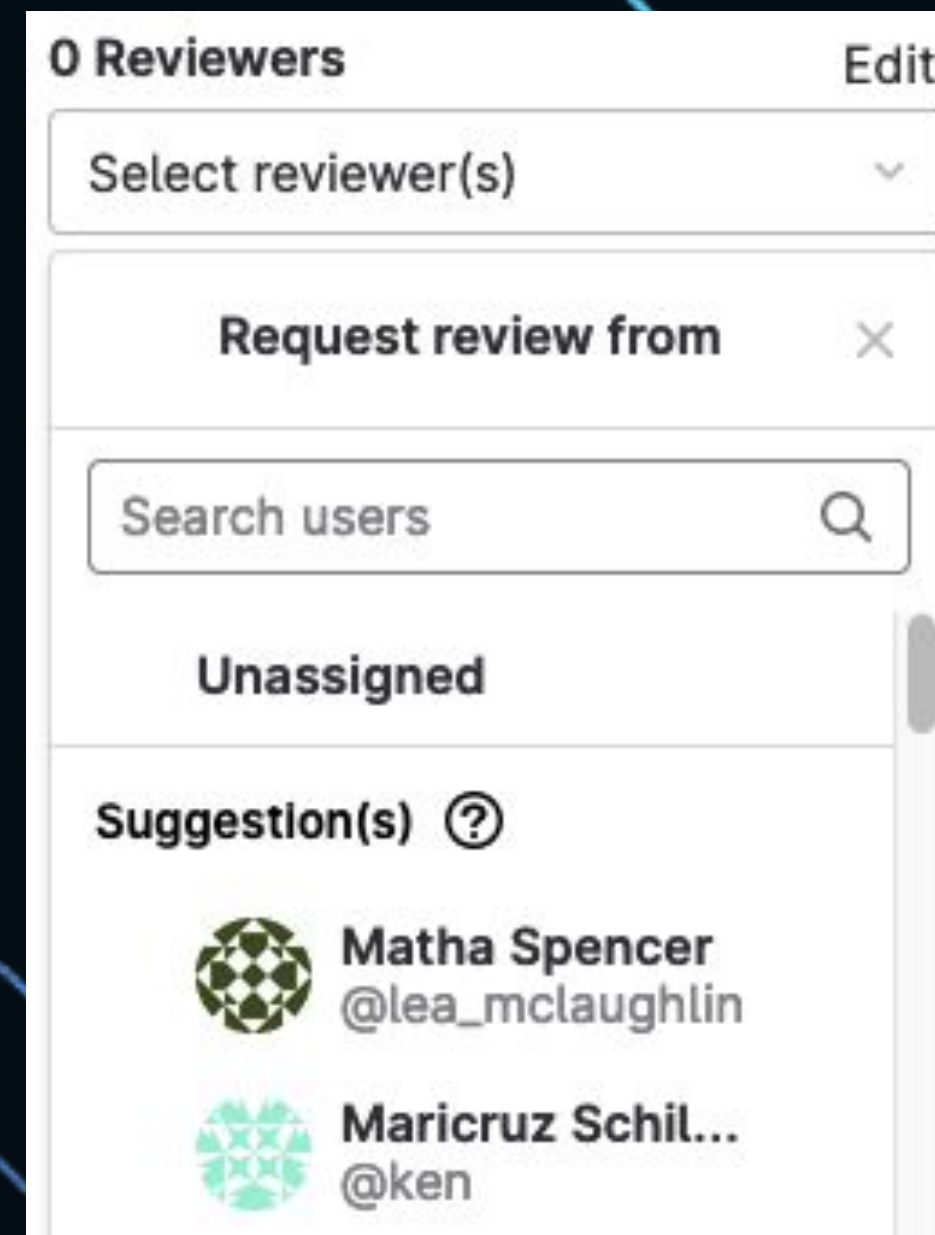
GitLabのAI機能の登場シーン

GitLabはDevSecOpsプラットフォームの名の通り、Value Streamの随所でAI機能が登場します。



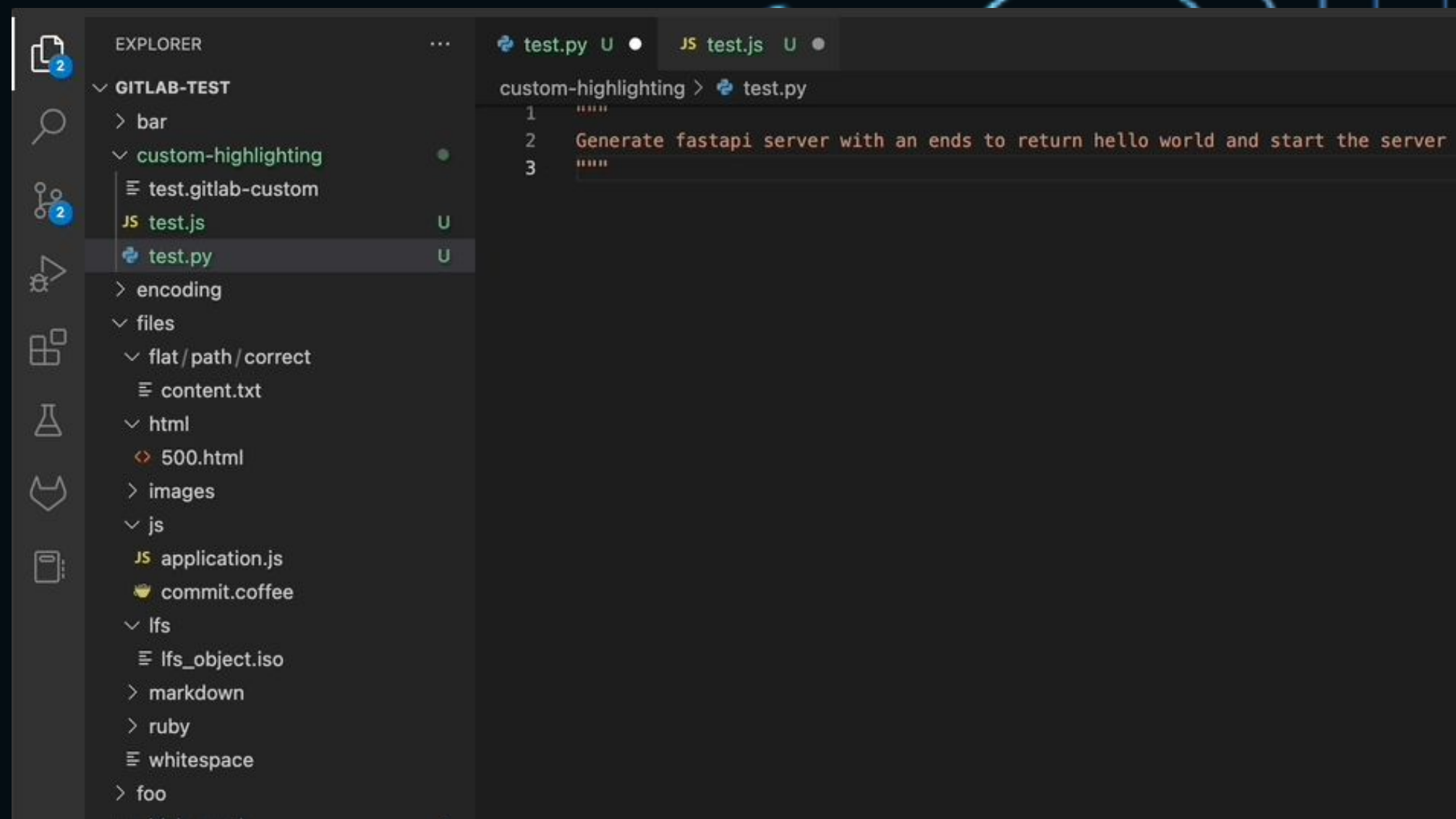
主なGitLabのAI機能

- Suggested Reviewers (GA)
 - マージリクエストの変更点とプロジェクトの貢献グラフを使って、機械学習によるレビュアーのサジェストが右サイダーのレビュアーセクションに表示されます



主なGitLabのAI機能

- Code Suggestions (Beta)
 - 生成AIがコードを提案します
 - 対応している言語
 - C++ / C# / Go / Google SQL / Java / JavaScript / Kotlin / PHP / Python / Ruby / Rust / Scala / Swift / TypeScript
 - 対応しているIDE
 - GitLab WebIDE / VS Code / JetBrains IDEs / Visual Studio / Neovim



主なGitLabのAI機能

- Explain this vulnerability (Beta)

- 大規模な言語モデルを使用することで、脆弱性の要約や修正案を提案し、解決を支援します

The screenshot displays the GitLab Duo interface for explaining a vulnerability. The main content area shows a code snippet with a vulnerability report for 'Improper Neutralization of Special Elements used in an SQL Command'. The report includes details such as Severity (Critical), Project (GitLab.org / security-products / Tests / WebGoat.NET), Tool (SAST), Scanner (Semgrep), and Location (File: WebGoat/App_Code/DB/MySqlCommandProvider.cs:304). It also lists identifiers like security_code_scan.SCS0002-1, CWE-89, and SCS0002. A section titled 'Explain this vulnerability and how to mitigate it with AI (Beta)' provides a description of the feature and a 'Try it out' button. Below this, there are 'Linked items' and a search bar. On the right side, a sidebar titled 'Explain this vulnerability (Beta)' shows the AI-generated response, including a 'Vulnerability Explanation' and an 'Attack Example' with a sample URL: `username=test&password=1234' or '1'='1`. At the bottom of the sidebar, there are buttons for 'Helpful', 'Unhelpful', and 'Wrong', along with a 'Submit' button.

主なGitLabのAI機能

- Explain this code (Experiment)

- AIがコードを説明することで、開発者が素早く理解できます

GitLab.org > GitLab > Repository

Code owners: James Lopez, Peter Leitzen, Douglas Barbosa Alexandre, 161 more

code_reuse_helpers.rb 5.35 KiB

2 Select the icon `ing_literal: true`

```
3 require 'forwardable'
4
5 require_relative '../lib/gitlab_edition'
6
7 module RuboCop
8   module CodeReuseHel
9     extend Forwardable
10
11     def_delegators :GitlabEdition, :ee?, :jh?
12
13     # Returns true for a `(send const ...)` node.
14     def send_to_constant?(node)
15       node.type == :send && node.children&.first&.type
16     end
17
18     # Returns `true` if the name of the receiving constant
19     # `String`.
20     def send_receiver_name_ends_with?(node, suffix)
21       return false unless send_to_constant?(node)
22
23       receiver_name = name_of_receiver(node)
24
25       receiver_name != suffix &&
26         receiver_name.end_with?(suffix)
27     end
28
29     # Returns the file path (as a `String`) for an AST
30     def file_path_for_node(node)
31       node.location.expression.source_buffer.name
32     end
33
34     # Returns the name of a constant node.
35     #
```

1 Highlight the code

Code Explanation Experiment

Responses generated by AI

You are not allowed to copy any part of this output into issues, comments, GitLab source code, commit messages, merge requests or any other user interface in the /gitlab-org or /gitlab-com groups.

Read more

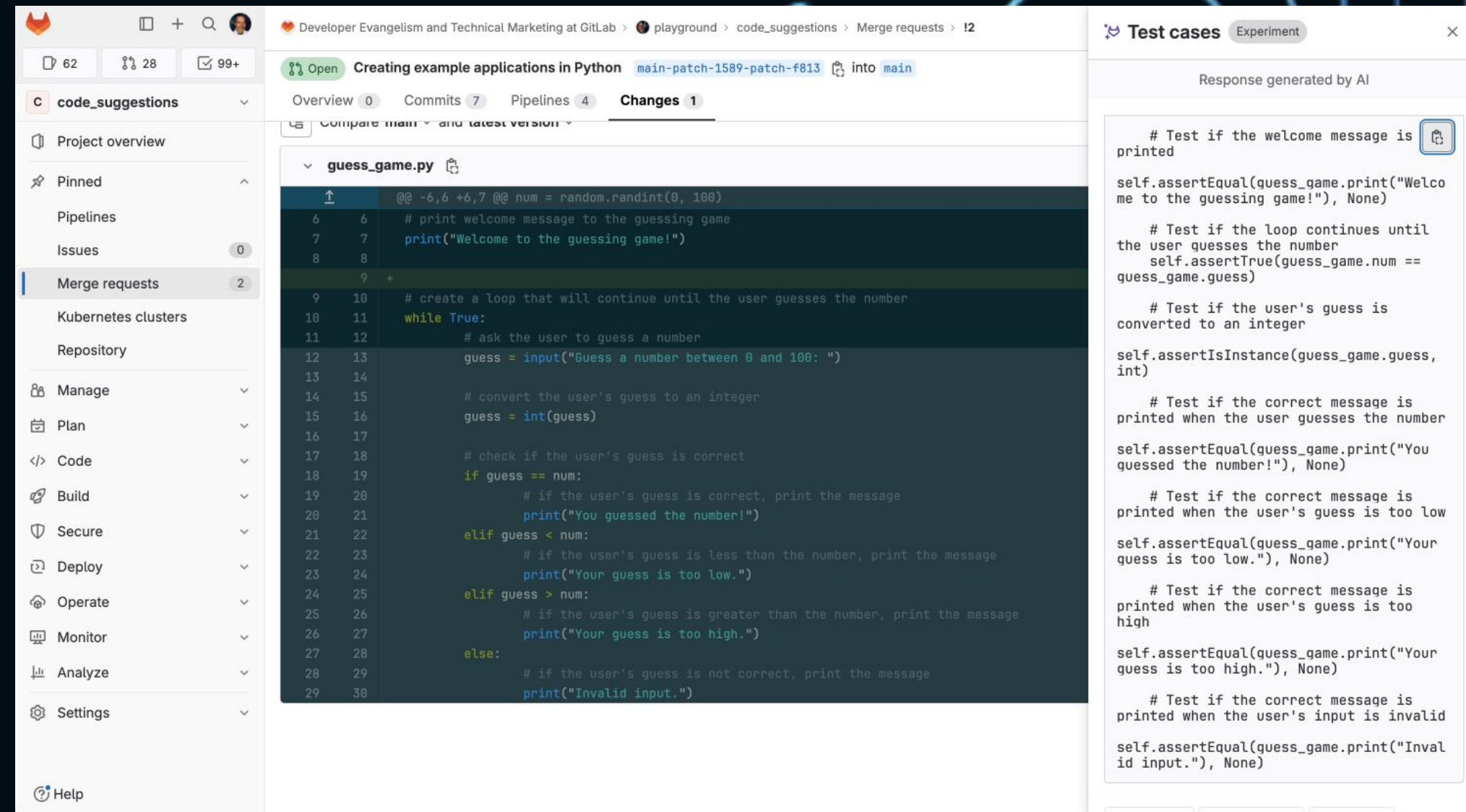
3 The code is explained

4 Give feedback

Helpful Unhelpful Wrong

主なGitLabのAI機能

- Generate Tests in Merge Request (Experiment)
 - マージリクエストでテストを提案します



主なGitLabのAI機能

- Issue summaries (Experiment)
 - イシューのディスカッションを要約します

The screenshot shows a GitLab issue page for issue #197239, titled "Reduce Secure Scanners root container executions in favor of unprivileged user mode". The issue was created 3 years ago by user rossfuhrman (Developer). The AI-generated summary includes a "Goal" section and a checklist of containers that run as root.

GitLab.org > GitLab > Issues > #197239

Open Issue created 3 years ago by [rossfuhrman](#) (Developer) Edit Close issue

Reduce Secure Scanners root container executions in favor of unprivileged user mode

Goal

Best practice is to not run containers as root. Especially with privileged containers and mounts on root filesystem.

This also means our secure containers are unable to run on OpenShift (see [#2068](#)) which does not allow root execution.

By reducing permissions we should likely enable sudoer access to still allow users to install dependencies; i.e. `before_script: apt install gcc`

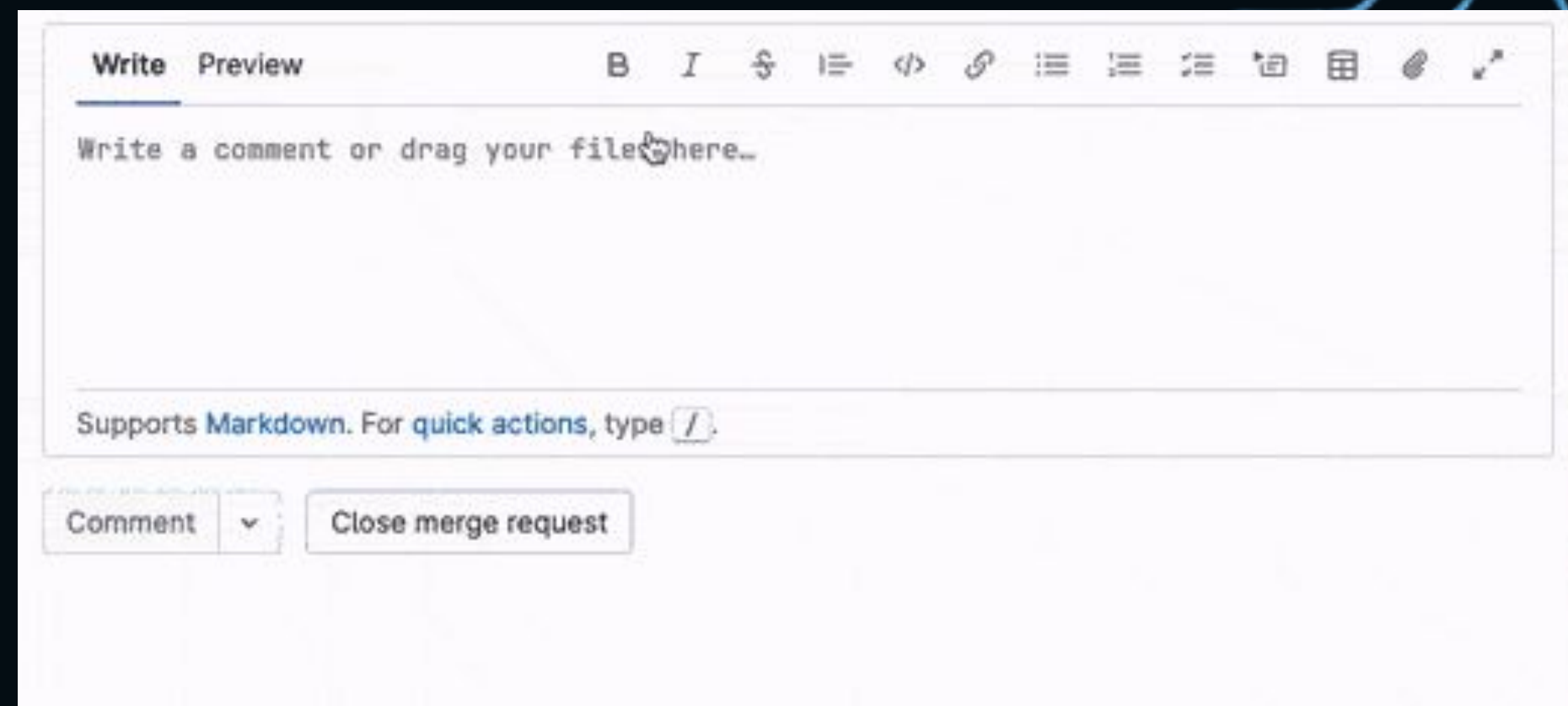
Containers that run as root

- bandit root
- brakeman root
- ~~bundler-audit-root~~ (#281816 (closed))
- flawfinder root
- gemnasium-root (#281816 (closed))
- gemnasium-maven-root (#281816 (closed))
- gemnasium-python-root (#281816 (closed))
- gosec root
- gcs root (#273530 (closed))
- pmd-apex root
- retire.js root (#281816 (closed))
- secrets root
- security-code-scan root
- sobelow root
- spotbugs root

0 of 15 checklist items completed · Edited 2 years ago by Sashi Kumar Kumaresan

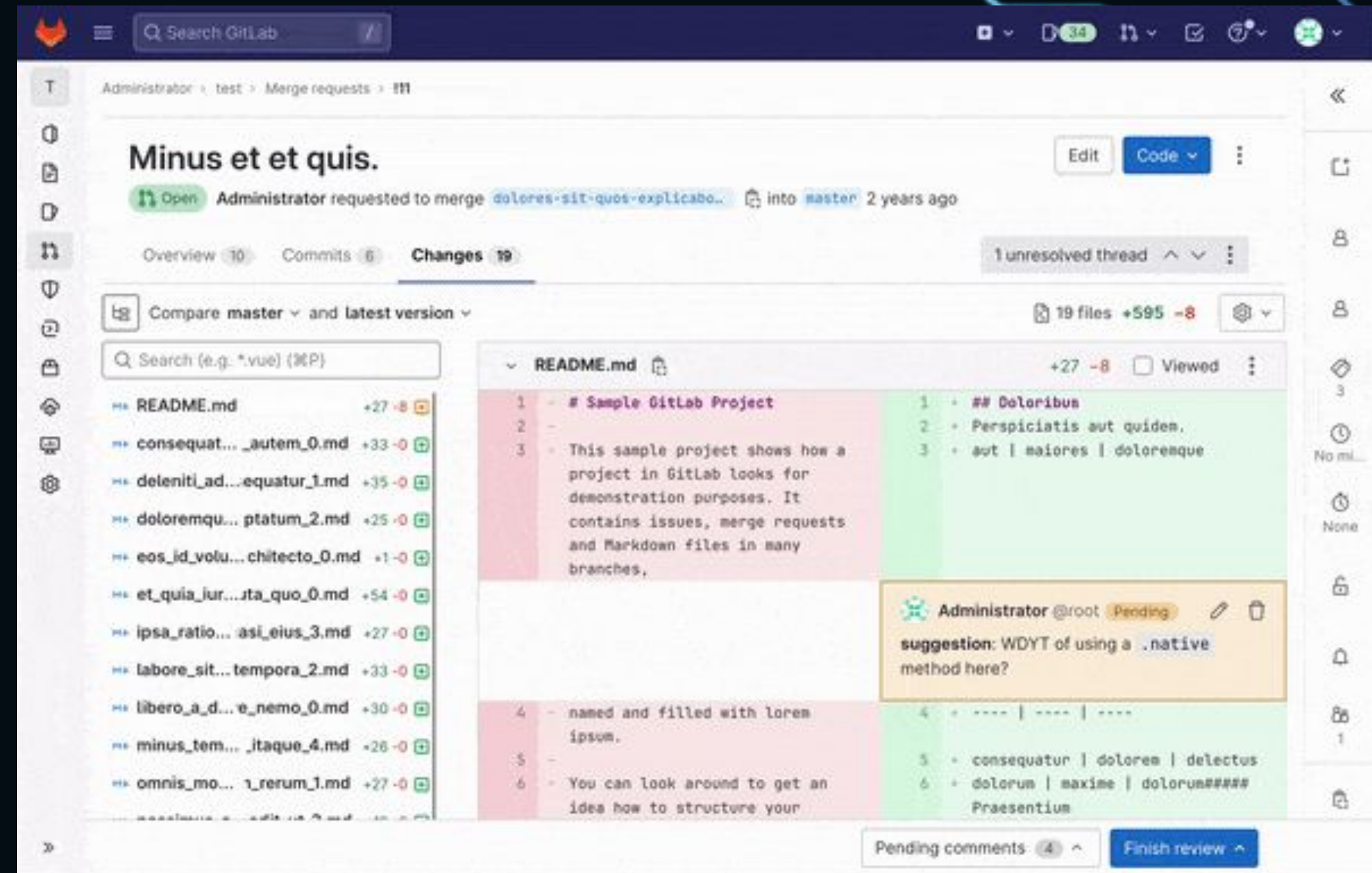
主なGitLabのAI機能

- Summarize Merge Request Changes (Experiment)
 - マージリクエストに含まれる変更点を要約します



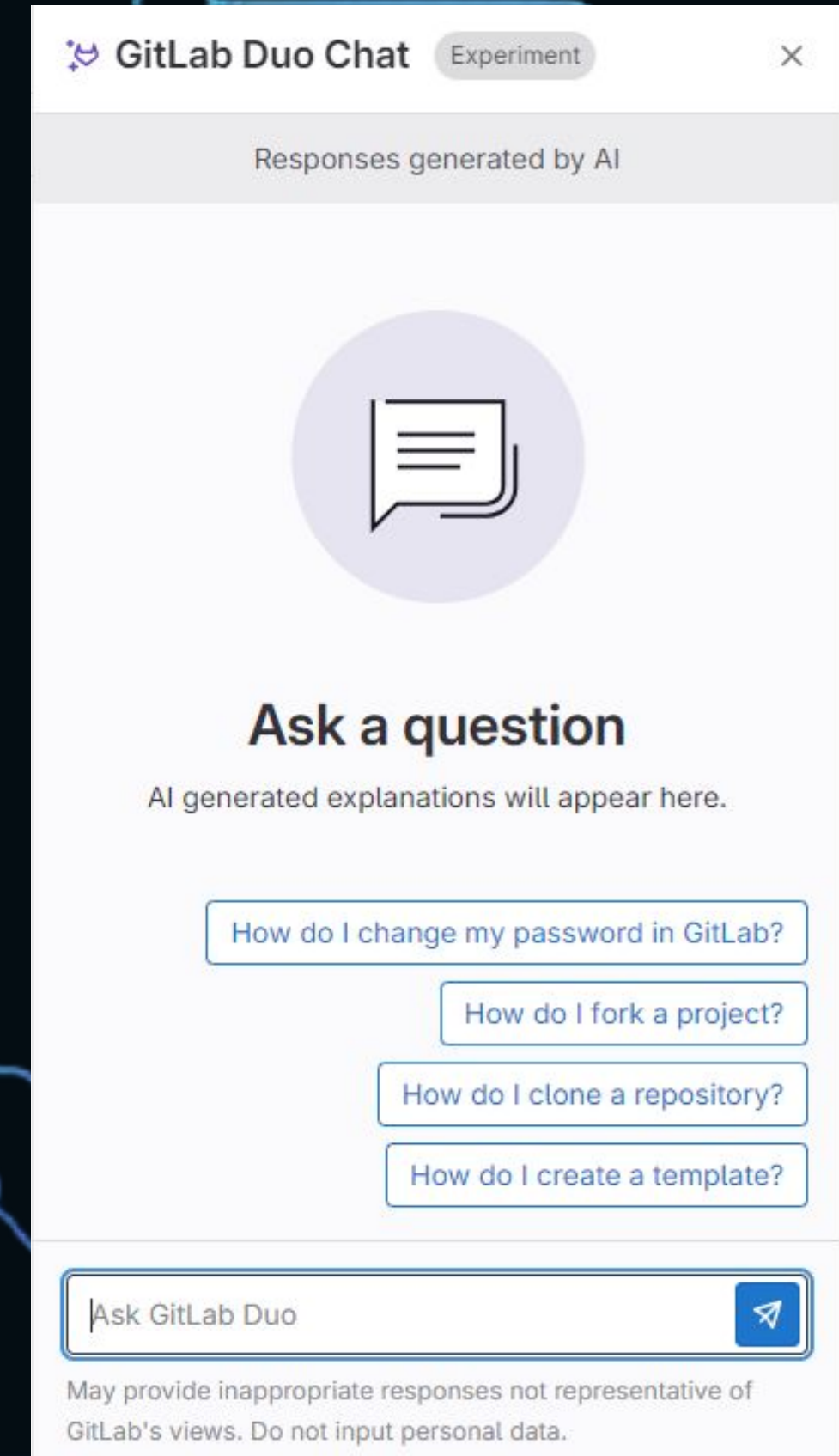
主なGitLabのAI機能

- Summarize My Merge Request Review (Experiment)
 - レビューアーの指摘を要約します



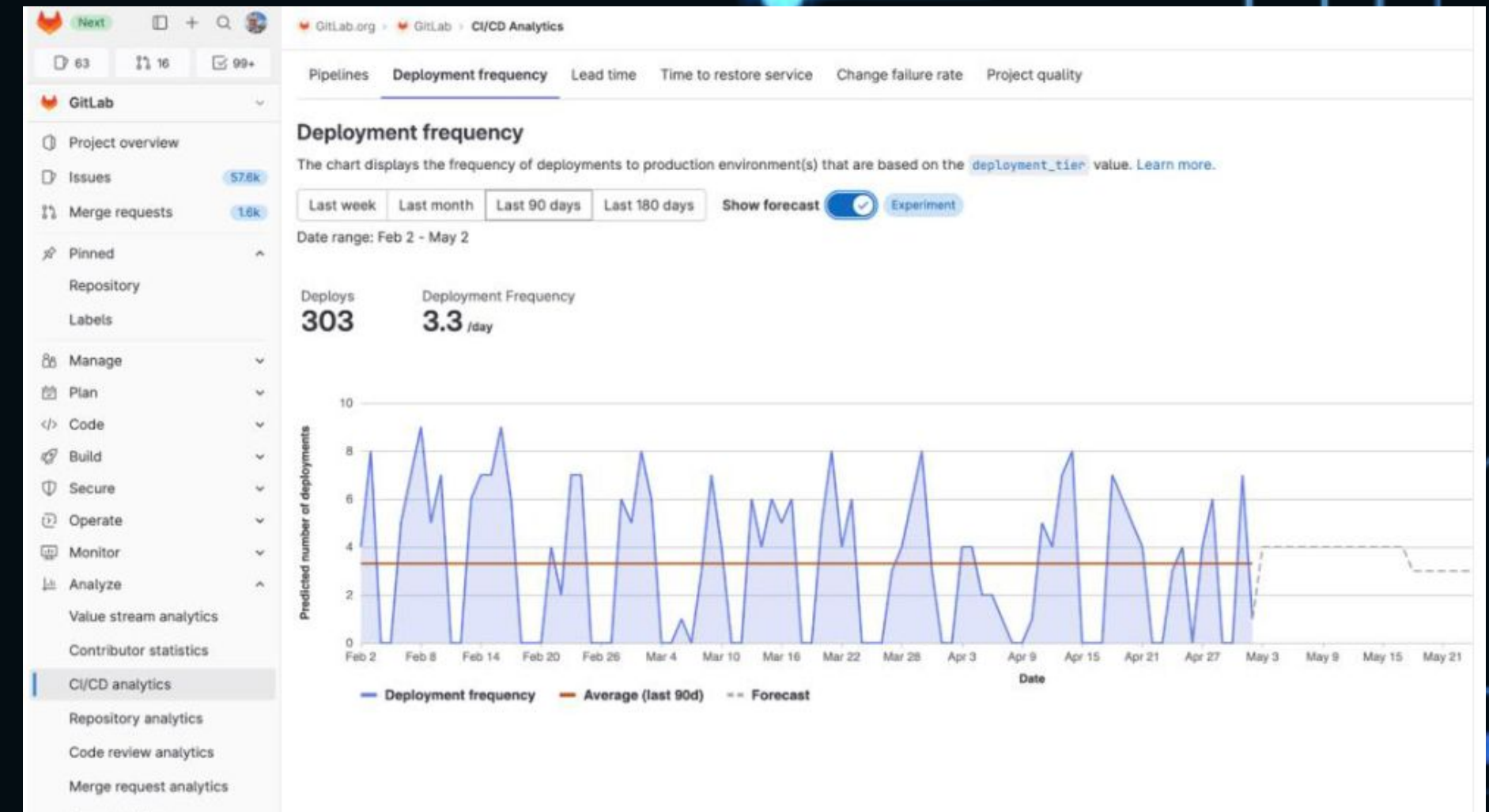
主なGitLabのAI機能

- GitLab Chat (Experiment)
 - 膨大な情報から素早く質問に答えます



主なGitLabのAI機能

- Value streams forecasting
(Experiment)
 - 過去の傾向からValue stream metricsを予測します



まとめ

GitLabはソフトウェア開発ライフサイクルのすべてのステージをカバーするDevSecOpsプラットフォーム

GitLabで開発すれば、自然とValue Stream分析に必要な情報が揃い、測定方法のカスタマイズが可能

あらゆるシーンで、生成AIが開発のお手伝いをしてくれる



THANK YOU!

ありがとうございました