



This presentation and linked pages contain information related to upcoming products, features, and functionality.

It is important to note that the information presented is for informational purposes only. Please do not rely on this information for purchasing or planning purposes.

As with all projects, the items mentioned in this presentation and linked pages are subject to change or delay. The development, release, and timing of any products, features, or functionality remain at the sole discretion of GitLab Inc.

Gaining efficiency with a little help from AI

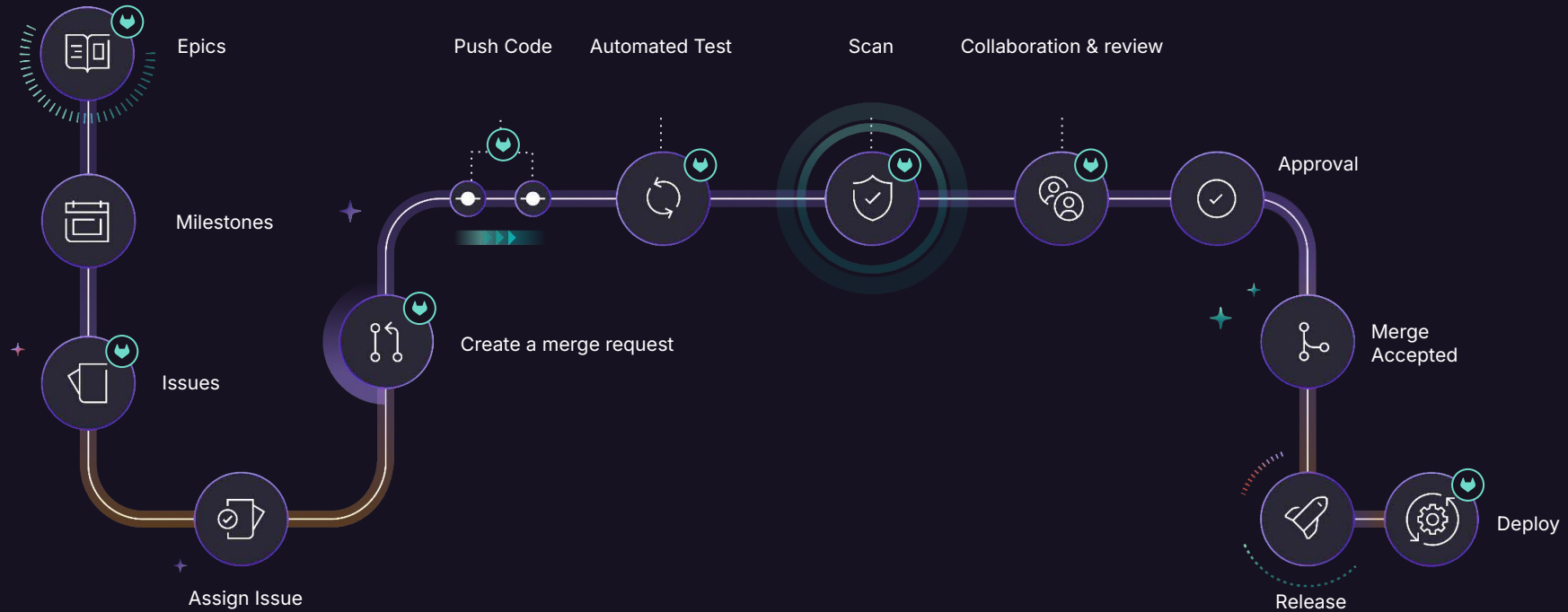
Gen-AI Coding Assistant Programme: GitLab - Duo
21 Feb 2025

Pedric Kng

Senior Solution Architect
@pedrickng



Which is the most inefficient task?



Get started

Your first assignment



Summarize the assignment

The issue discussion or description is long, and the due date is soon.

Example:

<https://gitlab.com/gitlab-org/gitlab/-/issues/423459>

GitLab.org > GitLab > Issues > #424338

Open Rich Text Editor should respect the `r` shortcut

Activity View summary Sort or filter

AI-generated summary Experiment

Incorrect behavior of "r" keyboard shortcut in rich text editor

- The "r" keyboard shortcut is not working in the rich text editor.
- This is a recent break.
- The plain text editor can be used as a workaround.
- This issue is similar to #408631, but not a duplicate.
- @vermainfinite is working on a fix.

Only visible to you · [Leave feedback](#)

John Hope added group knowledge bug functional type bug devops plan severity 3 scoped labels 1 month ago

John Hope added Category:Content Editor frontend labels 1 month ago

GitLab Bot added section dev scoped label 1 month ago

Assignee Edit
Anant Verma

Epic Edit
None

Labels Edit
SLO Near Miss bug functional devops plan group knowledge priority 3 section dev severity 3 type bug workflow in dev Category:Content Editor Community contribution frontend quick win

Milestone Edit
16.6



Lets get hacking



Development road blockers

How to clone a Git repository, and work with the CLI or IDE.

What is the syntax for adding CI/CD configuration?

Project templates, .gitignore, LICENSE, README.md ?



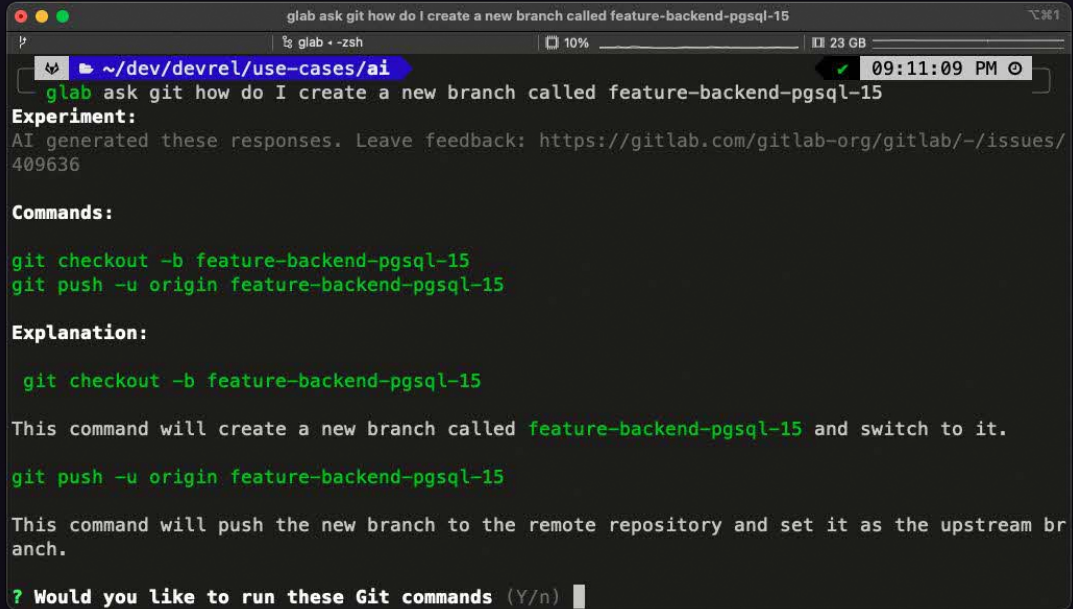
Development using Git

Can you remember the syntax for

1. Checking out a branch
2. Rebasing a branch
3. Tagging a release

The GitLab CLI can help:

`glab ask git <my question>`



```
glab ask git how do I create a new branch called feature-backend-pgsql-15
~/dev/devrel/use-cases/ai
glab ask git how do I create a new branch called feature-backend-pgsql-15
Experiment:
AI generated these responses. Leave feedback: https://gitlab.com/gitlab-org/gitlab/-/issues/409636

Commands:

git checkout -b feature-backend-pgsql-15
git push -u origin feature-backend-pgsql-15

Explanation:

git checkout -b feature-backend-pgsql-15

This command will create a new branch called feature-backend-pgsql-15 and switch to it.

git push -u origin feature-backend-pgsql-15

This command will push the new branch to the remote repository and set it as the upstream branch.

? Would you like to run these Git commands (Y/n)
```



Conversational Chat



Knowledgeable in

- GitLab documentation
- General programming and coding



Context awareness (UI)

- Epics
- Issues
- Code files
- Merge requests
- Commits
- Pipeline jobs

The screenshot displays the GitLab Duo Chat interface within a web browser. The left sidebar shows the GitLab navigation menu with 'Duo Chat' highlighted. The main content area shows a chat window titled 'Duo Chat Project Getting Started' with a project ID of 52619925. The chat content includes a welcome message, a list of project files (README.md, LICENSE, CHANGELOG, CONTRIBUTING, etc.), and a table of project files. A green arrow points from the 'Help' option in the sidebar to the chat window. Another green arrow points from the 'README.md' file in the chat window to the 'Project Structure' section on the right. The right sidebar contains two sections: 'Project Structure' and 'Best Practices', both with bulleted lists of recommendations. At the bottom right, there is a small GitLab Duo Chat logo and a disclaimer: 'May provide inappropriate responses that are not representative of GitLab's views. Do not input personal data.'



Code efficiency



New algorithms, design patterns, architecture, ... language features

As a senior developer, you feel confident in designing and implementing robust architectures.

Learning the latest programming language features requires time.

As a beginner in development, it is hard to understand specific algorithms, and know the documentation for structures, interfaces, etc.

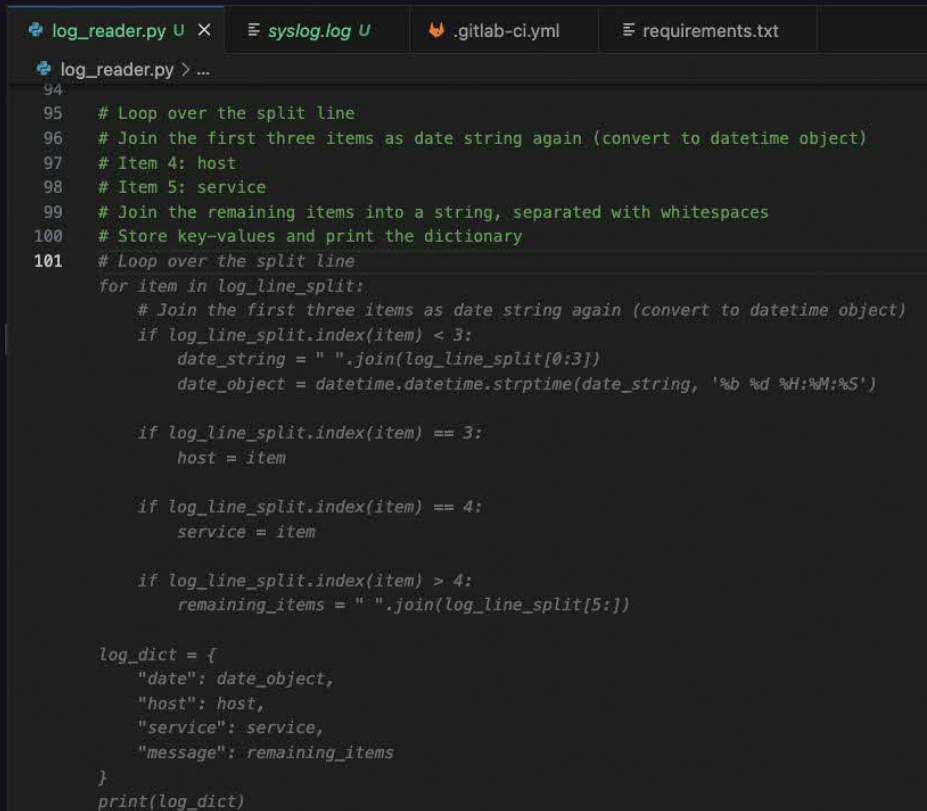
Learning under pressure and with many roadblocks ahead can lead to errors.



Efficient pair programming

Get context specific Code Suggestions

Provide instructions in detailed code comments, that serve as documentation too.



The screenshot shows a code editor with a dark theme. At the top, there are tabs for 'log_reader.py', 'syslog.log', '.gitlab-ci.yml', and 'requirements.txt'. The active tab is 'log_reader.py'. The code is a Python script that processes log lines. It includes several comments explaining the logic, such as 'Loop over the split line', 'Join the first three items as date string again (convert to datetime object)', and 'Store key-values and print the dictionary'. The code uses a loop to iterate over 'log_line_split' and constructs a dictionary 'log_dict' with keys 'date', 'host', 'service', and 'message'. The dictionary is then printed. The code is well-commented and includes detailed instructions that serve as documentation.

```
log_reader.py > ...
94
95 # Loop over the split line
96 # Join the first three items as date string again (convert to datetime object)
97 # Item 4: host
98 # Item 5: service
99 # Join the remaining items into a string, separated with whitespaces
100 # Store key-values and print the dictionary
101 # Loop over the split line
    for item in log_line_split:
        # Join the first three items as date string again (convert to datetime object)
        if log_line_split.index(item) < 3:
            date_string = " ".join(log_line_split[0:3])
            date_object = datetime.datetime.strptime(date_string, '%b %d %H:%M:%S')

        if log_line_split.index(item) == 3:
            host = item

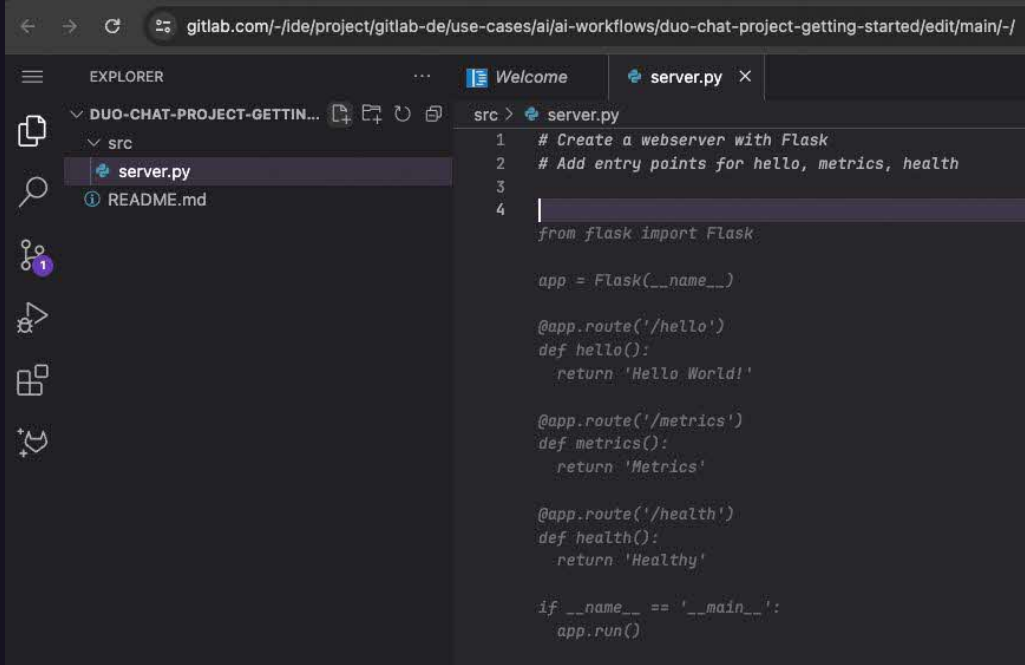
        if log_line_split.index(item) == 4:
            service = item

        if log_line_split.index(item) > 4:
            remaining_items = " ".join(log_line_split[5:])

    log_dict = {
        "date": date_object,
        "host": host,
        "service": service,
        "message": remaining_items
    }
    print(log_dict)
```



GitLab Duo Code Suggestions, Python



```
1 # Create a webserver with Flask
2 # Add entry points for hello, metrics, health
3
4
5 from flask import Flask
6
7 app = Flask(__name__)
8
9 @app.route('/hello')
10 def hello():
11     return 'Hello World!'
12
13 @app.route('/metrics')
14 def metrics():
15     return 'Metrics'
16
17 @app.route('/health')
18 def health():
19     return 'Healthy'
20
21 if __name__ == '__main__':
22     app.run()
```

```
220 # Create a pretty print function with indent 4
221 import json
222
223 def pretty_print(results):
224     print(json.dumps(results, indent=4))
225
226 pretty_print(results)
```

```
25 # Read all file paths in the directory into a list
26 import os
27
28 # Read all file paths in the directory into a list
29 log_files = []
30 for root, directories, files in os.walk(path):
31     for file in files:
32         if file.endswith(file_suffix):
33             log_files.append(os.path.join(root, file))
```

https://gitlab.com/gitlab-com/www-gitlab-com/-/merge_requests/130337/diffs



Best practices



Multi-line comments

1. Overall summary and problem statement
2. Refinements and specific implementation requirements
3. Describe expected functionality
4. Additional prompts to include dependencies or language specific requirements ("main" package in Go)

```
// Create a client for Kubernetes observability
// Create a function that
// Read the kubernetes configuration file from the KUBECONFIG env var
// Create kubernetes context, namespace default
// Inspect container, pod, service status and print an overview
// Import necessary packages
// Create main package
```



Context awareness

- Open file
- Filename and extension
- [Files open in tabs](#)
- Dependencies (via [Repository X-Ray](#))

GitLab > Duo Code Suggestions: **Open Tabs Context**

☒ Use the contents of open tabs as context

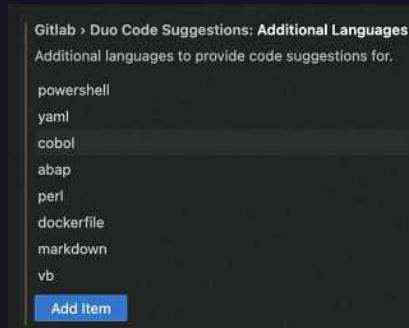


Tips: Additional languages



Code Suggestions support tested languages from the underlying LLM

- But additional languages can be configured for use



Gitlab > Duo Code Suggestions: Enabled Supported Languages

Enable Code Suggestions for these languages.

- ☒ C (c)
- ☒ C++ (cpp)
- ☒ C# (csharp)
- ☒ Go (go)
- ☒ HAML (haml)
- ☒ Handlebars (handlebars)
- ☒ Java (java)
- ☒ JavaScript (javascript)
- ☒ JavaScript React (javascriptreact)
- ☒ Kotlin (kotlin)
- ☒ Python (python)
- ☒ PHP (php)
- ☒ Ruby (ruby)
- ☒ Rust (rust)
- ☒ Scala (scala)
- ☒ Shell (shellscript)
- ☒ SQL (sql)
- ☒ Swift (swift)
- ☒ TypeScript (typescript)
- ☒ TypeScript React (typescriptreact)
- ☒ Svelte (svelte)
- ☒ Terraform (terraform)
- ☒ Terragrunt (terragrunt)
- ☒ Vue (vue)



Duo Chat in IDE

During implementation



What does this code do?

Imagine the following situation:

Joined a new team, maintaining a new code base,
with weekly bug fix sprints.

Spotted an algorithm with no code comments.

No unit tests.

No documentation.

How long does it take to search the internet, write
tests, intentionally break the code ...

Spoiler: 1 day, week, month.

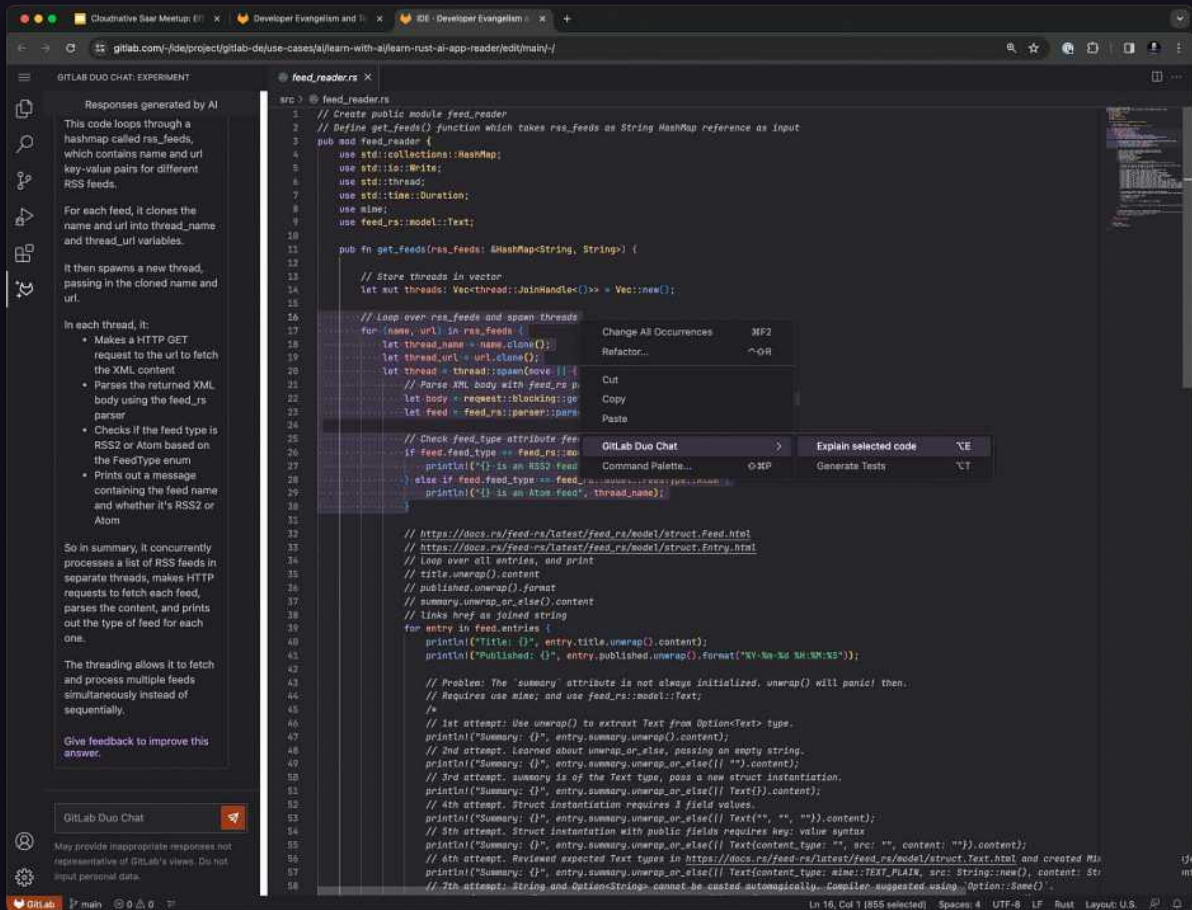


Explain this code

Learning while coding.

Integrated into your IDE,
avoiding context
switches

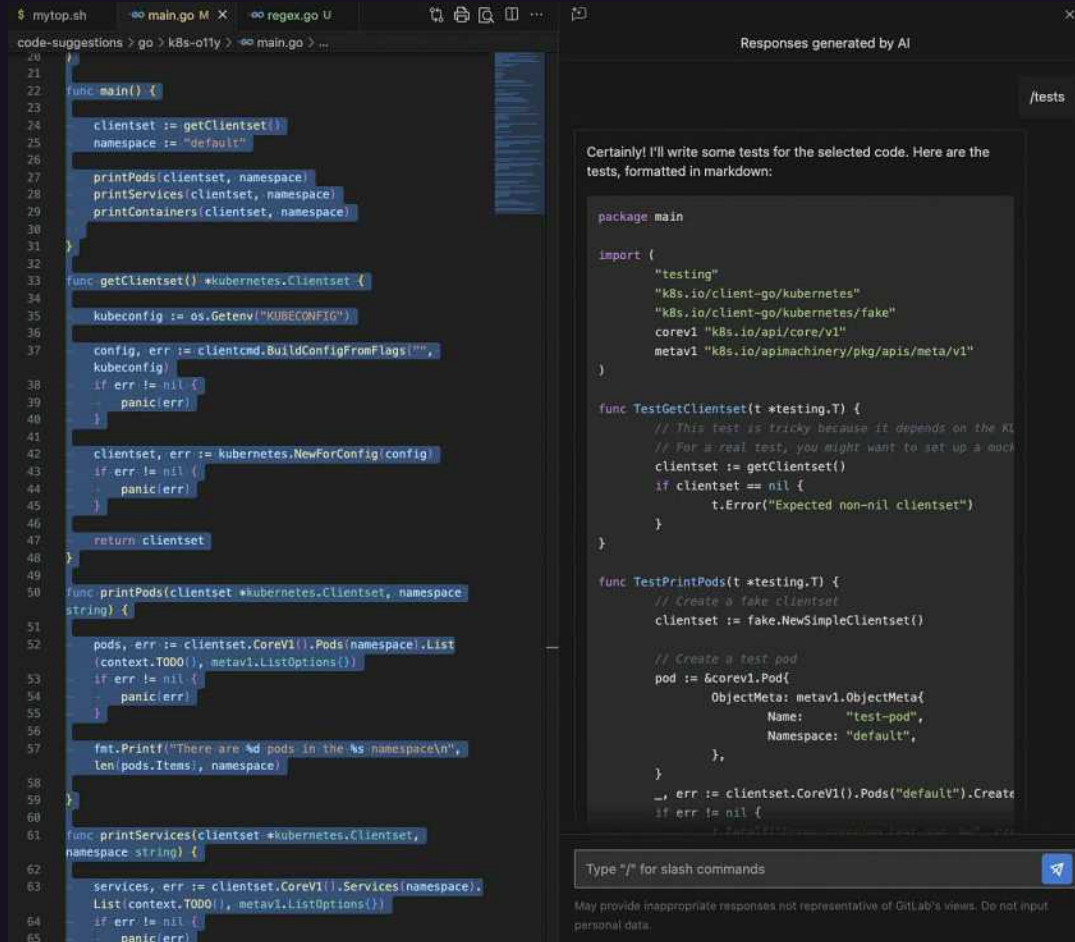
- `/explain the performance`
- `/explain focus on the algorithm`
- `/explain the performance gains or losses using this code`
- `/explain the object inheritance (classes, object-oriented)`
- `/explain why a static variable is used here (C++)`
- `/explain how this function would cause a segmentation fault (C)`
- `/explain how concurrency works in this context (Go)`
- `/explain how the request reaches the client (REST API, database)`



Test Generation

Generate tests with AI to avoid regressions

- Use a specific test framework, for example `/tests` using the Boost.test framework (C++) or `/tests` using Jest (JavaScript).
- Focus on extreme test cases, for example `/tests` focus on extreme cases, force regression testing.
- Focus on performance, for example `/tests` focus on performance.
- Focus on regressions and potential exploits, for example `/tests` focus on regressions and potential exploits.



The screenshot shows a code editor with two panes. The left pane displays Go code for a Kubernetes client, and the right pane shows tests generated by AI.

Left Pane (main.go):

```
20
21
22 func main() {
23
24     clientset := getClientset()
25     namespace := "default"
26
27     printPods(clientset, namespace)
28     printServices(clientset, namespace)
29     printContainers(clientset, namespace)
30
31 }
32
33 func getClientset() *kubernetes.Clientset {
34
35     kubeconfig := os.Getenv("KUBECONFIG")
36
37     config, err := clientcmd.BuildConfigFromFlags("",
38         kubeconfig)
39     if err != nil {
40         panic(err)
41     }
42
43     clientset, err := kubernetes.NewForConfig(config)
44     if err != nil {
45         panic(err)
46     }
47
48     return clientset
49 }
50
51 func printPods(clientset *kubernetes.Clientset, namespace
52 string) {
53
54     pods, err := clientset.CoreV1().Pods(namespace).List
55     (context.TODO(), metav1.ListOptions{})
56     if err != nil {
57         panic(err)
58     }
59
60     fmt.Printf("There are %d pods in the %s namespace\n",
61         len(pods.Items), namespace)
62
63 }
64
65 func printServices(clientset *kubernetes.Clientset,
66 namespace string) {
67
68     services, err := clientset.CoreV1().Services(namespace).
69     List(context.TODO(), metav1.ListOptions{})
70     if err != nil {
71         panic(err)
72     }
73 }
```

Right Pane (Responses generated by AI):

```
package main

import (
    "testing"
    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/kubernetes/fake"
    corev1 "k8s.io/api/core/v1"
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

func TestGetClientset(t *testing.T) {
    // This test is tricky because it depends on the K1
    // For a real test, you might want to set up a mock
    clientset := getClientset()
    if clientset == nil {
        t.Error("Expected non-nil clientset")
    }
}

func TestPrintPods(t *testing.T) {
    // Create a fake clientset
    clientset := fake.NewSimpleClientset()

    // Create a test pod
    pod := &corev1.Pod{
        ObjectMeta: metav1.ObjectMeta{
            Name:      "test-pod",
            Namespace: "default",
        },
    },
    _, err := clientset.CoreV1().Pods("default").Create
    if err != nil {
        t.Errorf("Expected no error creating pod")
    }
}
```

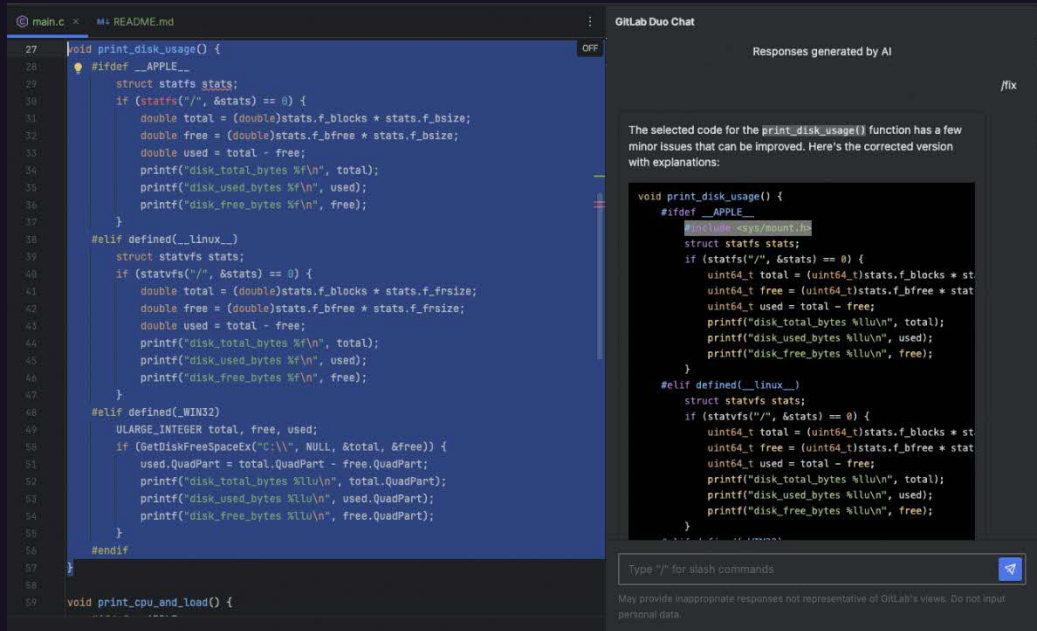
Below the code, there is a search bar with the text "Type '/' for slash commands" and a search icon. At the bottom, a disclaimer reads: "May provide inappropriate responses not representative of GitHub's views. Do not input personal data."



Fix code

Identify bugs and get fix suggestions

- Focus on grammar and typos, for example, `/fix grammar mistakes and typos.`
- Focus on a concrete algorithm or problem description, for example, `/fix duplicate database inserts or /fix race conditions.`
- Focus on potential bugs that are not directly visible, for example, `/fix potential bugs.`
- Focus on code performance problems, for example, `/fix performance problems.`
- Focus on fixing the build when the code does not compile, for example, `/fix the build.`



The screenshot displays a code editor with a C function `print_disk_usage()` and a chat window on the right. The code in the editor is as follows:

```
27 void print_disk_usage() {
28     #ifdef __APPLE__
29         struct statfs stats;
30         if (statfs("/", &stats) == 0) {
31             double total = (double)stats.f_blocks * stats.f_bsize;
32             double free = (double)stats.f_bfree * stats.f_bsize;
33             double used = total - free;
34             printf("disk_total_bytes %f\n", total);
35             printf("disk_used_bytes %f\n", used);
36             printf("disk_free_bytes %f\n", free);
37         }
38     #elif defined(__linux__)
39         struct statvfs stats;
40         if (statvfs("/", &stats) == 0) {
41             double total = (double)stats.f_blocks * stats.f_frsize;
42             double free = (double)stats.f_bfree * stats.f_frsize;
43             double used = total - free;
44             printf("disk_total_bytes %f\n", total);
45             printf("disk_used_bytes %f\n", used);
46             printf("disk_free_bytes %f\n", free);
47         }
48     #elif defined(_WIN32)
49         ULARGE_INTEGER total, free, used;
50         if (GetDiskFreeSpaceEx("C:\\", NULL, &total, &free)) {
51             used.QuadPart = total.QuadPart - free.QuadPart;
52             printf("disk_total_bytes %llu\n", total.QuadPart);
53             printf("disk_used_bytes %llu\n", used.QuadPart);
54             printf("disk_free_bytes %llu\n", free.QuadPart);
55         }
56     #endif
57 }
58
59 void print_cpu_and_load() {
```

The chat window on the right, titled "Responses generated by AI", shows a suggestion for the `print_disk_usage()` function. The suggestion includes a corrected version of the code that uses `uint64_t` for integer values and includes `<sys/mount.h>` for the `statfs` function on Apple systems.

The selected code for the `print_disk_usage()` function has a few minor issues that can be improved. Here's the corrected version with explanations:

```
void print_disk_usage() {
    #ifdef __APPLE__
        #include <sys/mount.h>
        struct statfs stats;
        if (statfs("/", &stats) == 0) {
            uint64_t total = (uint64_t)stats.f_blocks * st
            uint64_t free = (uint64_t)stats.f_bfree * stat
            uint64_t used = total - free;
            printf("disk_total_bytes %llu\n", total);
            printf("disk_used_bytes %llu\n", used);
            printf("disk_free_bytes %llu\n", free);
        }
    #elif defined(__linux__)
        struct statvfs stats;
        if (statvfs("/", &stats) == 0) {
            uint64_t total = (uint64_t)stats.f_blocks * st
            uint64_t free = (uint64_t)stats.f_bfree * stat
            uint64_t used = total - free;
            printf("disk_total_bytes %llu\n", total);
            printf("disk_used_bytes %llu\n", used);
            printf("disk_free_bytes %llu\n", free);
        }
    }
}
```

Type "/" for slash commands

May provide inappropriate responses not representative of GitHub's views. Do not input personal data.



Code Refactoring

Improve code quality,
refactor code with AI

- Use a specific coding pattern, for example `/refactor with ActiveRecord` or `/refactor into a class providing static functions`.
- Use a specific library, for example `/refactor using mysql`.
- Use a specific function/algorithm, for example `/refactor into a stringstream` with multiple lines in C++.
- Refactor to a different programming language, for example `/refactor to TypeScript`.
- Focus on performance, for example `/refactor improving performance`.
- Focus on potential vulnerabilities, for example `/refactor avoiding memory leaks and exploits`.

The screenshot shows a code editor with two tabs: `main.go` and `regex.go`. The `main.go` tab is active, displaying a Go program that interacts with the Kubernetes API. The code includes functions for getting a clientset, printing pods, services, and containers. The right sidebar shows the 'Responses generated by AI' for the command `/refactor improve readability`. The response lists five suggestions for improving readability and provides the refactored code. The refactored code uses more descriptive comments, consistent error handling, and extracts common functionality into a separate function.

```
code-suggestions > go > k8s-otly > ~go main.go > ...
```

```
20
21
22 func main() {
23
24     clientset := getClientset()
25     namespace := "default"
26
27     printPods(clientset, namespace)
28     printServices(clientset, namespace)
29     printContainers(clientset, namespace)
30
31 }
32
33 func getClientset() *kubernetes.Clientset {
34
35     kubeconfig := os.Getenv("KUBECONFIG")
36
37     config, err := clientcmd.BuildConfigFromFlags("",
38         kubeconfig)
39     if err != nil {
40         panic(err)
41     }
42
43     clientset, err := kubernetes.NewForConfig(config)
44     if err != nil {
45         panic(err)
46     }
47
48     return clientset
49 }
50
51 func printPods(clientset *kubernetes.Clientset, namespace
52 string) {
53
54     pods, err := clientset.CoreV1().Pods(namespace).List
55     (context.TODO(), metav1.ListOptions{})
56     if err != nil {
57         panic(err)
58     }
59
60     fmt.Printf("There are %d pods in the %s namespace\n",
61         len(pods.Items), namespace)
62
63 }
64
65 func printServices(clientset *kubernetes.Clientset,
66 namespace string) {
67
68     services, err := clientset.CoreV1().Services(namespace).
69     List(context.TODO(), metav1.ListOptions{})
70     if err != nil {
71         panic(err)
72     }
73
74     fmt.Printf("There are %d services in the %s namespace\n",
75         len(services.Items), namespace)
76
77 }
```

Responses generated by AI

/refactor improve readability

To improve readability, I'll make the following changes:

1. Add more descriptive comments
2. Use consistent error handling
3. Extract common functionality into a separate function
4. Improve variable naming
5. Use constants for repeated values

Here's the refactored code:

```
package main

import (
    "context"
    "fmt"
    "os"

    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/tools/clientcmd"
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

const {
    defaultNamespace = "default"
}

func main() {
    clientset, err := getClientset()
    if err != nil {
        handleError("Failed to create Kubernetes cl")
    }

    printResourceCount(clientset, defaultNamespace, "po")
    printResourceCount(clientset, defaultNamespace, "se")
    printResourceCount(clientset, defaultNamespace, "cc")
}
```

Type "/" for slash commands

May provide inappropriate responses not representative of GitHub's views. Do not input personal data.



Duo Chat /slash cmd

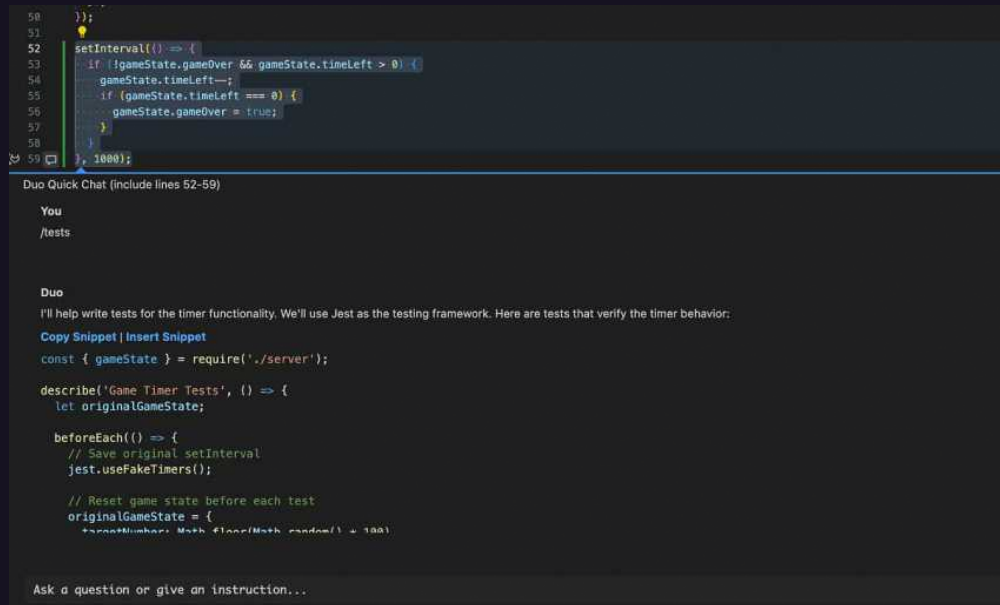


Command	Purpose
/clear	Delete all conversations permanently and clear the chat window
/reset	Start a new conversation, but keep the previous conversations visible in the chat window
/help	Learn more about how Duo Chat works
/tests	Write tests
/explain	Explain code
/refactor	Refactor the code
/fix	Fix the code
/include	Include file context



Quick Duo Chat

- In-editor window chat
- Code selection context awareness
- Keyboard shortcut
 - Mac OS: **Option + c**
 - Win & Linux: **Alt + c**



Ready for review



MR Summaries

Inside the Merge Request (everyone)

The screenshot shows a GitLab Merge Request (MR) interface. On the left is a sidebar with navigation links: Project (cli), Pinned, Issues (6k), Merge requests (23), Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, and Analyze. The main content area displays the MR details for 'test: Fix test Test_repoArchive_Integration integration test', which is merged and requested by Jay McCure. It includes an 'Overview' tab, a 'Latest AI-generated summary' section, a 'Description' section, and a right-hand sidebar with fields for Assignee (Jay McCure), Reviewer (Michael Friedrich), Labels (devops, code review, maintenance, pipelines, section dev, type maintenance, Category: GitLab CLI, Quality), Milestone (None), Time tracking (No estimate or time spent), and 4 Participants.

test: Fix test Test_repoArchive_Integration integration test Edit Code

Merged Jay McCure requested to merge 1381-fix-replace-flaky-tes... into main 2 months ago

Overview 11 Commits 1 Pipelines 5 Changes 1

All threads resolved!

Latest AI-generated summary Experiment

This code change fixes a race condition in the Test_repoArchive_Integration integration test. The race condition occurs when another process has the binary that is being executed open for writing. This can happen when the binary is built and then executed to run the integration tests by using fork+exec. The fix is to sleep for a little bit to retry the command after the file lock has been released hopefully.

2 months ago · [Leave feedback](#)

Description

As observed in <https://gitlab.com/gitlab-org/cli/-/jobs/4785867150#L192>, this integration test has been failing occasionally. Logging was added which found that the root cause was the following error:

```
{ "Time": "2023-08-01T23:42:35.891151618Z", "Action": "output", "Package": "gitlab.
```

The issue is best summarised by <https://go.goglesource.com/go/+/go1.9.5/src/cmd/go/internal/work/build.go#2018> and this MR uses the same fix, which is basically to wait a short amount of time and retry if a `text file is busy` error occurs. Presumably this is happening because we are building the binary and then immediately executing a command against it.

An alternative solution would be to remove this test altogether, but with this MR we will be able to keep the test running.

Assignee Edit
Jay McCure

Reviewer Edit
Michael Friedrich

Labels Edit
devops create x
group code review x
maintenance pipelines x
section dev x type maintenance x
Category: GitLab CLI x Quality x
cli x

Milestone Edit
None

Time tracking +
No estimate or time spent

4 Participants
Jay McCure Michael Friedrich Jay McCure Jay McCure



Security

Explain and mitigate vulnerabilities



Security vulnerability

Imagine the following situation:

Security incident is going on.
Affects customers running the software.
Hot fix and long term fix is required.

What is a format string vulnerability, command injection, timing attack or buffer overflow?

How to fix the code without introducing (performance) regressions?



Vulnerability Explanation

- Get an understanding how malicious attackers exploit vulnerabilities.
- Better understand the why and how-to-fix with urgent security fixes.
- How: GitLab UI, chat prompt with follow-up questions.

Developer Evangelism and Technical Marketing at GitLab > ... > AI Workflows > Explain V

Needs triage Detected · 5 minutes ago in pipeline 1051085087

Potential time of check time of use vulnerability (open/fopen)

Description

Usage of the `open` family of functions may hint at a potential Time Of Check Time Of Use (TOCTOU) vulnerability. An attacker could exploit this by modifying the file being specified by the `open` function prior to the `open` function being called.

Prior to calling `open`, use `lstat` to open the file and confirm the attributes are correct. For example, use `fstat` on the `open` file descriptor to confirm that `st_dev` and `st_ino` are equal before calling `open` on the file's contents.

For more information please see: <https://wiki.sei.cmu.edu/confluence/display/c/FIO45-C.+Avoid+TOCTOU+race+conditions+while+accessing+files>

Severity: Info

Project: Developer Evangelism and Technical Marketing at GitLab / use-cases / AI / AI Vulnerability

Tool: SAST

Scanner: Semgrep

Location

File: `main.c:19`

```
19 fp = fopen("aipowered.devsecops.platform", "r");
```

Identifiers

- flawfinder.fopen-1.open-1
- A5:2017 - Broken Access Control
- [CWE-362](#)
- Flawfinder - fopen
- Flawfinder - open

Explain this vulnerability and how to mitigate it with AI Beta

This is a beta feature that uses AI to explain the vulnerability and provide recommendations. Please provide your feedback and ideas in [this issue](#).

☒ Send code with prompt 🔗

[Show prompt](#)

[Explain vulnerability](#)

Response generated by AI

Potential time of check time of use vulnerability (open/fopen)

Description

A potential time of check time of use (TOCTOU) vulnerability exists in the following code:

```
fp = fopen("aipowered.devsecops.platform", "r");
```

This code opens the file "aipowered.devsecops.platform" for reading. However, between the time that the file is opened and the time that it is read, an attacker could modify the file or delete it. This would cause the program to fail.

How an attacker can take advantage of the vulnerability

An attacker could take advantage of this vulnerability by creating a file named "aipowered.devsecops.platform" and then waiting for the program to open it. Once the program has opened the file, the attacker could modify the file or delete it. This would cause the program to fail.

How to fix the vulnerability

The vulnerability can be fixed by using the `O_NOFOLLOW` flag when opening the file. This flag prevents the program from following symbolic links, which would prevent an attacker from creating a symbolic link to a malicious file.

The following code shows how to fix the vulnerability:

```
fp = fopen("aipowered.devsecops.platform", "r", O_NOFOLLOW);
```

Vulnerability Resolution

- AI-driven resolution; create Merge request patch with changes
- Remember to review merge request and execute scan as AI guardrails

Resolve vulnerability: Potential format string vulnerability

Merge request actions ▾

 Open Michael Friedrich requested to merge `remediate/potential-format...` into `main` just now



Overview 0 Commits - Pipelines 1 **Changes 1**

 Compare `main` ▾ and `latest version` ▾

src/cmd.c 

+1 -1  Viewed  ⋮

↑

@@ -14,7 +14,7 @@ int main(int argc, char **argv) {

14 14

// Print the first argument to greet the caller -

https://owasp.org/www-community/attacks/Format_string_attack

16 16

printf("Hello");

17

- printf(argv[1]);

17

+ printf("%s", argv[1]);

18 18

// Pass the second argument to the system call -

https://owasp.org/www-community/attacks/Command_Injection

19 19

char *cat = "cat ";

20 20

↓

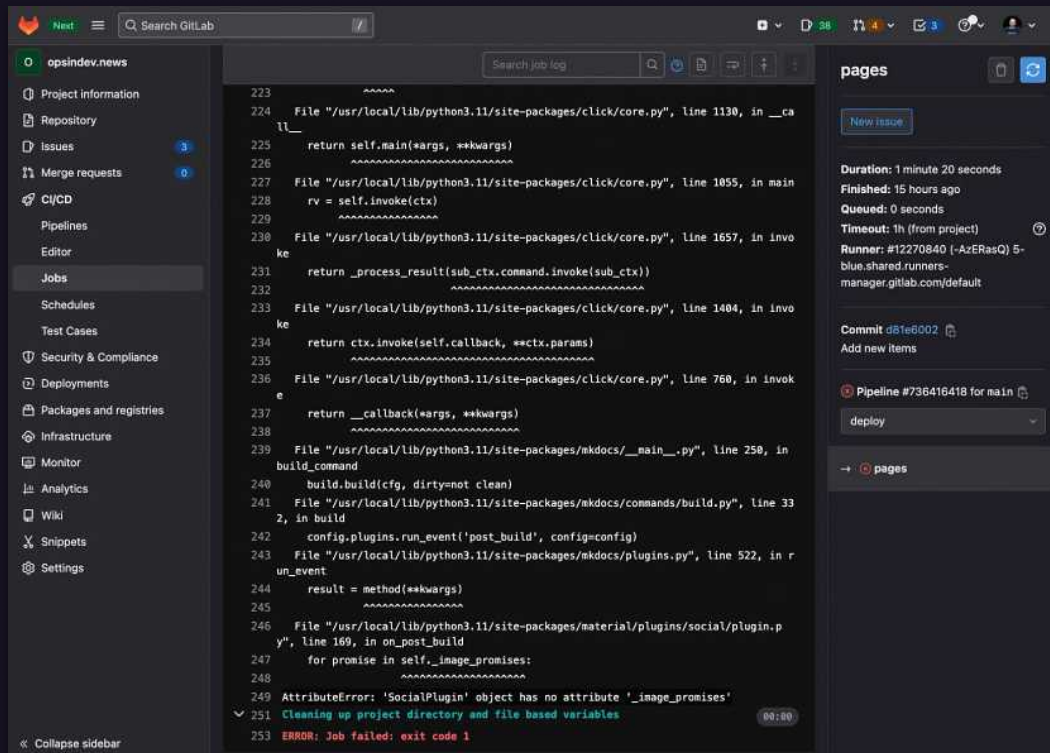


Debugging CI/CD



ETA to fix CI/CD?

A long stacktrace, endless scrolling until the error condition, greeted with a cryptic error message ...



Root cause analysis

Developer Evangelism and Technical Marketing at GitLab > ... > AI Workflows > Cid Root

Root cause analysis Search job log

```
19 Using docker image sha256:17e65561fd2c7ecae6297c2cc049960607ff94f
857f94569fc9c199 for python:latest with digest python@sha256:2586c
eb6673bc66d18f0a628a997c293b41268bc981e826bc0b5a92 ...
20 $ python main.py
21 Traceback (most recent call last):
22   File "/builds/gitlab-de/use-cases/ai/ai-workflows/cicd-root-cau
is-python/main.py", line 14, in <module>
23     main()
24   File "/builds/gitlab-de/use-cases/ai/ai-workflows/cicd-root-cau
is-python/main.py", line 9, in main
25     config = parse_config('config.ini')
26     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
27   File "/builds/gitlab-de/use-cases/ai/ai-workflows/cicd-root-cau
is-python/main.py", line 2, in parse_config
28     config = configparser.ConfigParser()
29     ^^^^^^^^^^^^^^^^^
30 NameError: name 'configparser' is not defined. Did you forget to
onfigparser'?
31 Cleaning up project directory and file based variables
32 ERROR: Job failed: exit code 1
```

Root cause analysis

The job failed because the `configparser` module is not installed. The error message states that `configparser` is not defined, which means that it is not available in the Python environment. This could be because the module was not installed, or because it was not installed in the correct location.

To fix the job, you could install the `configparser` module using the following command:

```
pip install configparser
```

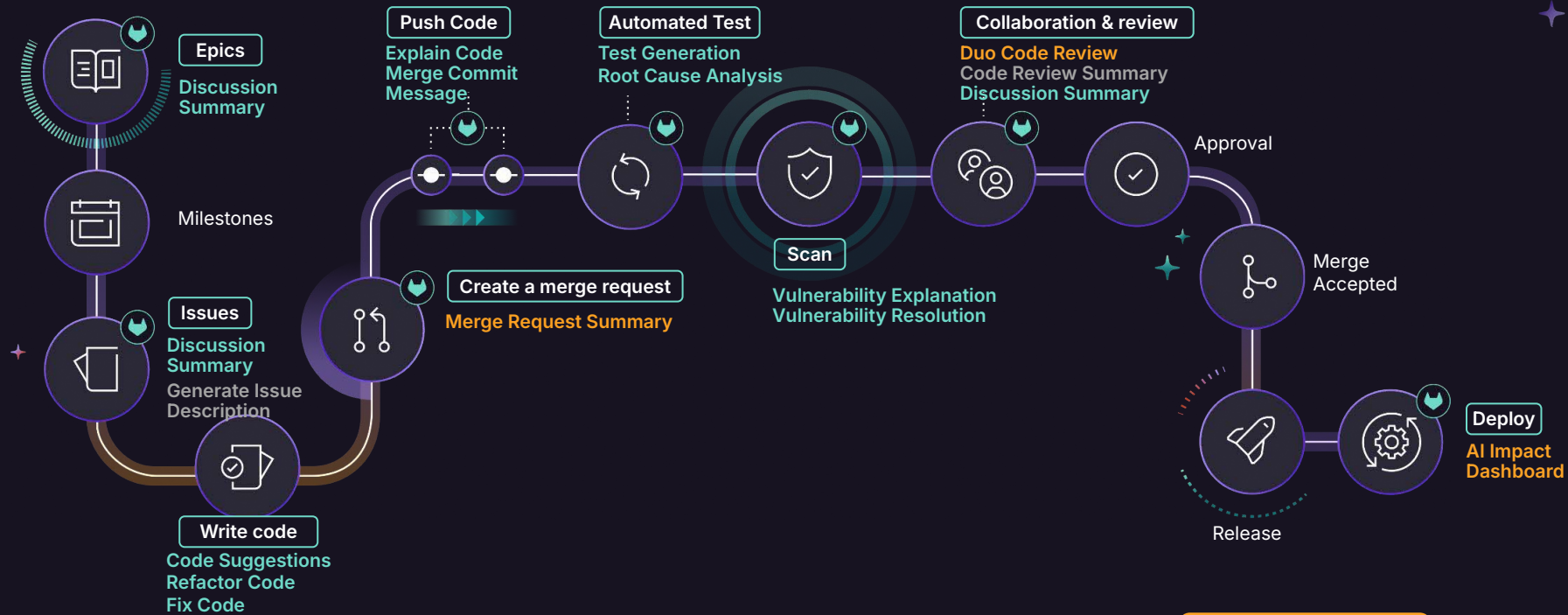
Once the module is installed, the job should be able to run successfully.

Example Fix:

```
# Import the configparser module
import configparser

# Parse the config file
config = configparser.ConfigParser()
config.read('config.ini')
```





Coming soon to
dedicated



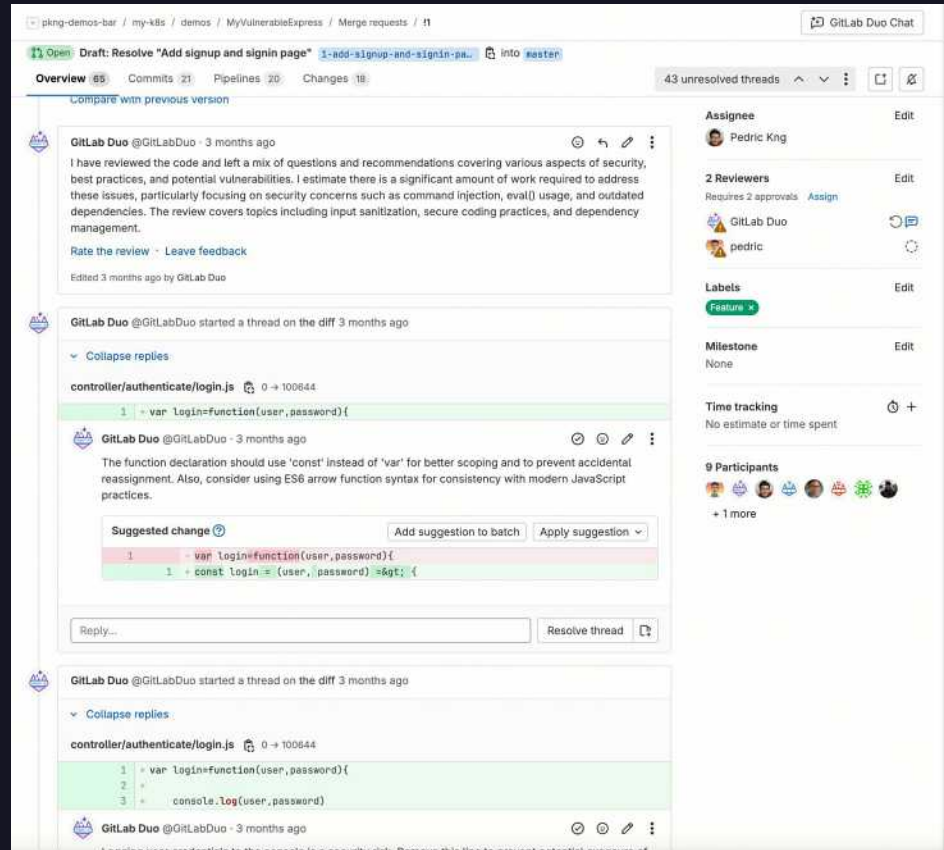
What's coming ?



Duo Code Review

Reduce wait time for reviewers, faster merges

- Assign Duo bot for review
- Accept suggestions
- Interact with Duo bot in reviews



Learning Resources

💡 GitLab University
<https://university.gitlab.com/pages/ai>

💡 Docs: GitLab Duo use cases:
https://docs.gitlab.com/ee/user/gitlab_duo/use_cases.html

💡 Blogs: What's new ! Best practices !
<https://about.gitlab.com/blog/categories/ai-ml/>



Thank you