The Complete Guide to Reviewing Pull Requests with PR Review Agent

PR Review Agent

Contents

1. Introduction **4**

2. Fundamentals **6**

3. Getting Started **9**

4. Features and Workflows **12**

4.1 Automatic Review 12

4.2 /describe 13

4.3 /review 14

4.4 /improve 15

4.5 /ask 18

4.6 /update_changelog 20

4.7 /add_docs 20

4.8 /generate_labels 21

5. Use Cases **23**

6. Tips, Patterns and Troubleshooting **26**

7. Resources and Reference **29**

CHAPTER 1

Introduction

1. Introduction

The PR Review Agent is an AI-powered pull request reviewer that integrates directly into GitHub, GitLab, Bitbucket, and Azure DevOps. The moment a pull request is opened, the agent triggers automatically, reads every changed file, analyzes the code against best practices, identifies security vulnerabilities and logic errors, generates a structured description of what the PR does, estimates the review effort required, and posts all of its findings as comments directly on the pull request, before any human reviewer has even opened it.

A pull request is the standard mechanism in version control for proposing code changes: a developer finishes work on a branch, then opens a pull request to merge those changes into the main codebase. The PR Review Agent participates in this process as an automated first reviewer that runs instantly on every PR, regardless of the time of day or the team's current workload. Code review is one of the most valuable and most expensive activities in software development; a thorough review of a non-trivial PR can take a senior engineer 30 to 60 minutes. The PR Review Agent handles the first layer of that review automatically, catching security vulnerabilities, race conditions, missing ownership checks, and code quality problems before a human ever opens the PR.

Who this guide is for

- Engineering teams who want a consistent, thorough first review on every pull request
- Senior engineers looking to reduce the time spent on routine PR review
- Teams that need security vulnerabilities caught before code reaches production
- Organizations evaluating an Enterprise, self-hostable code review solution

What you will learn

- What the PR Review Agent does automatically the moment a pull request is opened
- How to install it on GitHub and the other supported platforms
- How to use every manual command: /describe, /review, /improve, /ask, /update_changelog, /add_docs, and /generate_labels
- Common issues and how to resolve them

CHAPTER 2

Fundamentals

2. Fundamentals

What the PR Review Agent is

The PR Review Agent is not a linter and not a static analysis tool bolted onto a CI pipeline. It is an AI reviewer that reads a pull request the way a senior engineer would: understanding what changed, why it likely changed, and what could go wrong as a result. It integrates as an app on the version control platform itself, so its output appears exactly where a human reviewer's comments would.

The problem it solves

As teams grow, the review burden accumulates. A team of 10 developers opening 2 to 3 PRs per week generates a steady stream of reviews that get delayed, reviewers who get fatigued, and details that get missed. The PR Review Agent applies a consistent, thorough first review to every PR regardless of how many are open simultaneously, catching issues a tired or rushed human reviewer might miss every time.

Core concepts

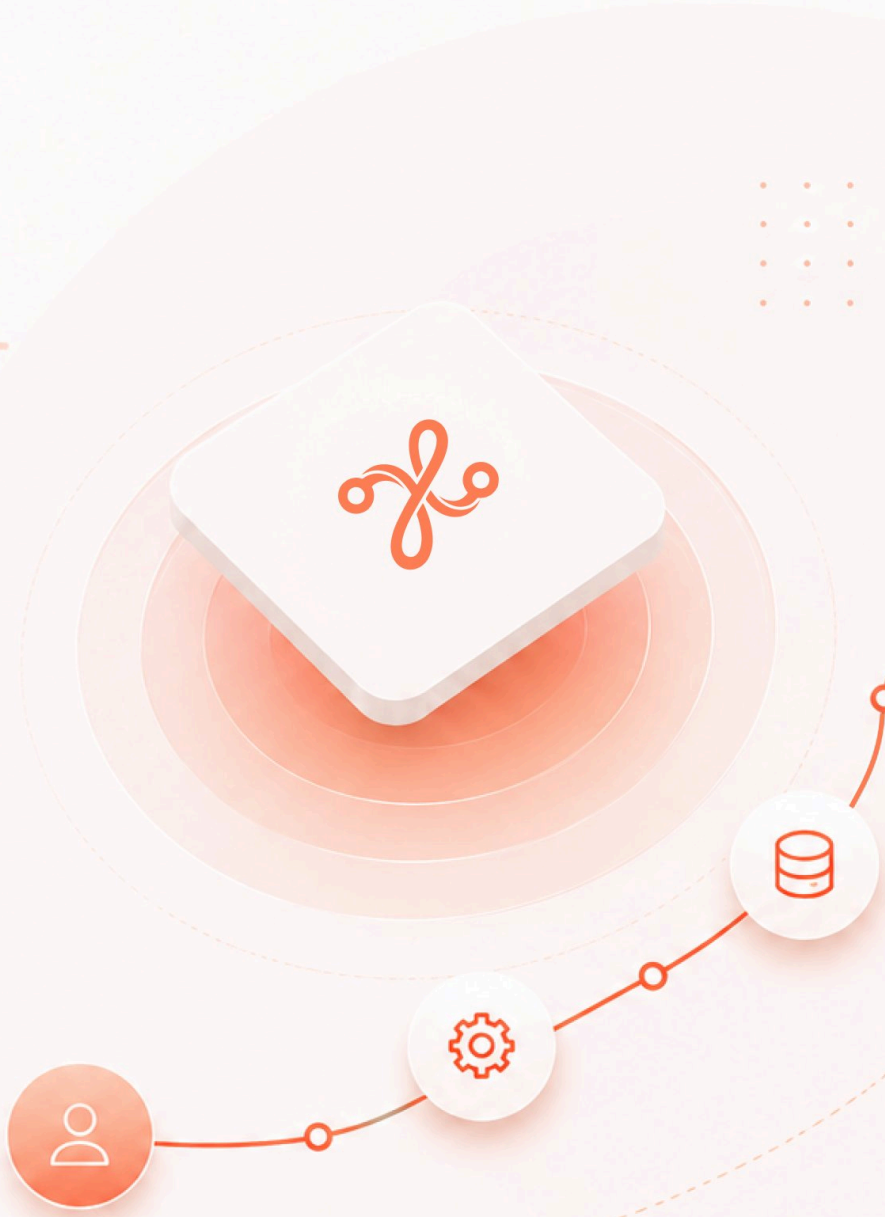
- **Automatic, instant review.** The agent triggers on every pull request event: opened, reopened, marked ready for review, or synchronized with a new commit. Most reviews complete in under a minute.
- **Seven manual commands.** Beyond the automatic review, the agent responds to commands typed as PR comments: /describe, /review, /improve, /ask, /update_changelog, /add_docs, and /generate_labels.
- **Effort estimation.** Every review includes a 1 to 5 review effort score, applied as a label, so a senior engineer knows at a glance whether a PR needs minutes or dedicated time.
- **Enterprise product.** The PR Review Agent is available exclusively on CodeMate's Enterprise plan, with a fully self-hosted deployment option for organizations with strict data residency or compliance requirements.

What runs automatically versus on request

- ✓ PR description, File Walkthrough, Reviewer Guide, effort score, and label, generated automatically on every pull request, no action required
- ✗ Concrete, ranked code suggestions with diffs, requires the /improve command
- ✗ Answers to free-text questions about the PR, requires the /ask command
- ✗ Changelog entries and inline documentation, require /update_changelog and /add_docs respectively

CHAPTER 3

Getting Started



3. Getting Started

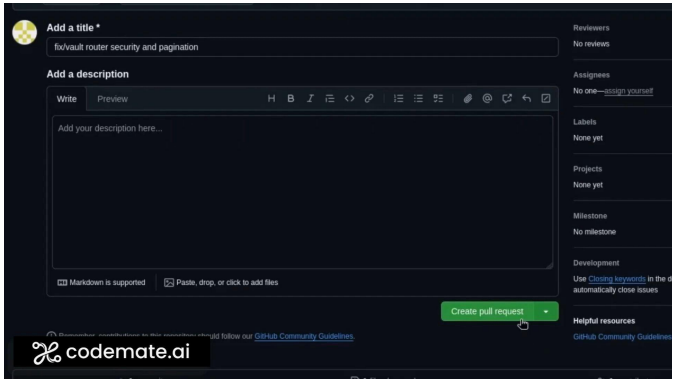
Installing the PR Review Agent

The PR Review Agent is installed as a GitHub App (with equivalent integrations for GitLab, Bitbucket, and Azure DevOps). Navigate to the CodeMate integrations page or the PR Review Agent app listing from the GitHub Marketplace, select the repositories to enable it on, and grant the required permissions. Installation is available exclusively to Enterprise plan organizations; contact contact@codemate.ai to arrange access and, if needed, a fully self-hosted deployment within your own infrastructure.

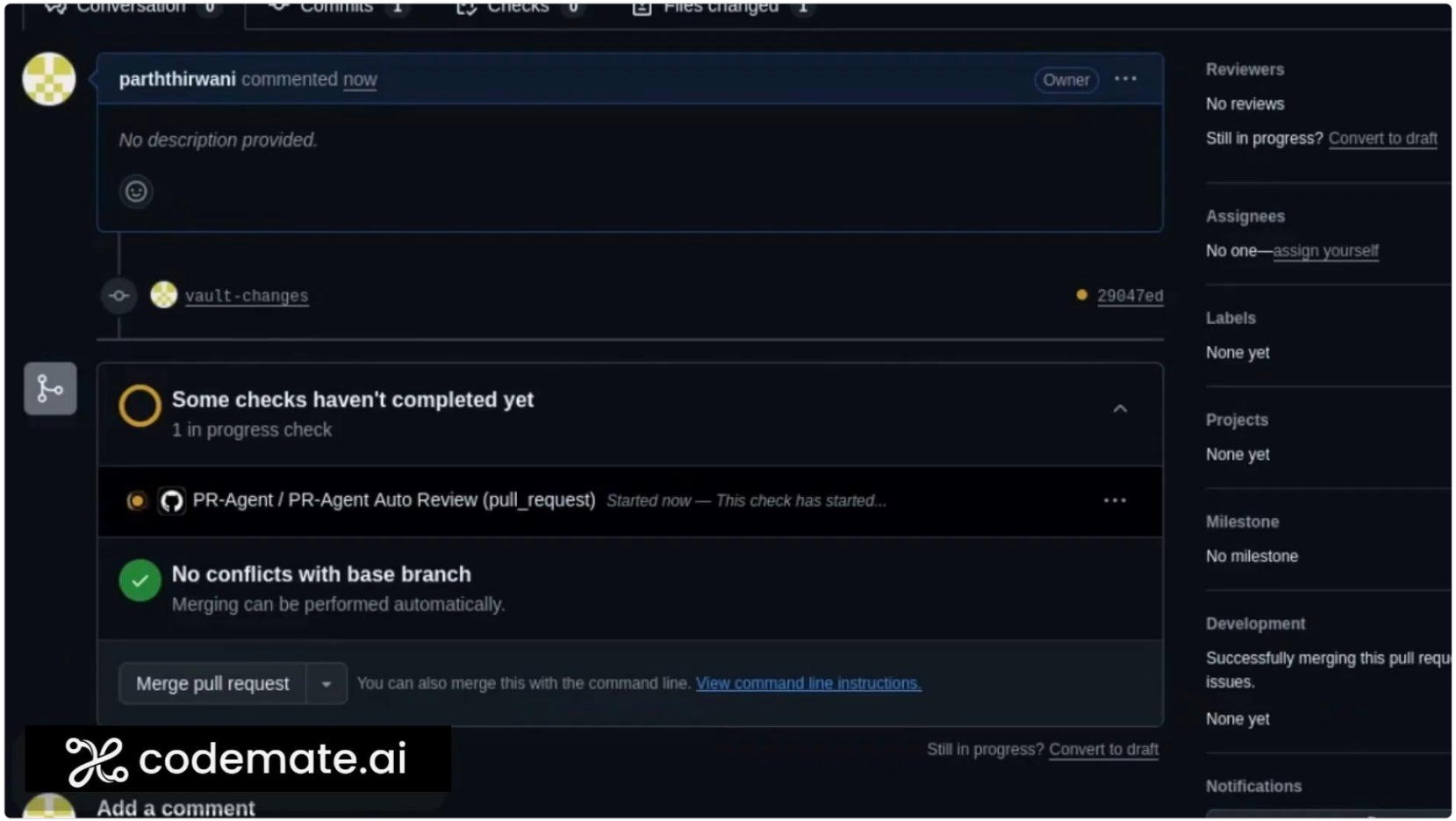
What happens when a pull request is opened

Once installation is complete, the agent is active immediately on the next pull request opened in any selected repository. The moment a PR is opened, the agent receives a webhook event and starts the PR-Agent Auto Review check. It fetches the diff, reads all changed files, runs its analysis in 30 to 60 seconds, then posts a structured description, a File Walkthrough, a PR Reviewer Guide with key observations, an effort estimation, and a corresponding label, all without any action from the developer.

> **Triggering manual commands**
>
> In addition to the automatic review, the agent responds to commands posted as PR comments. Type the command into the Add a comment box at the bottom of the PR conversation and click Comment: /describe, /review, /improve, /ask, /update_changelog, /add_docs, or /generate_labels.



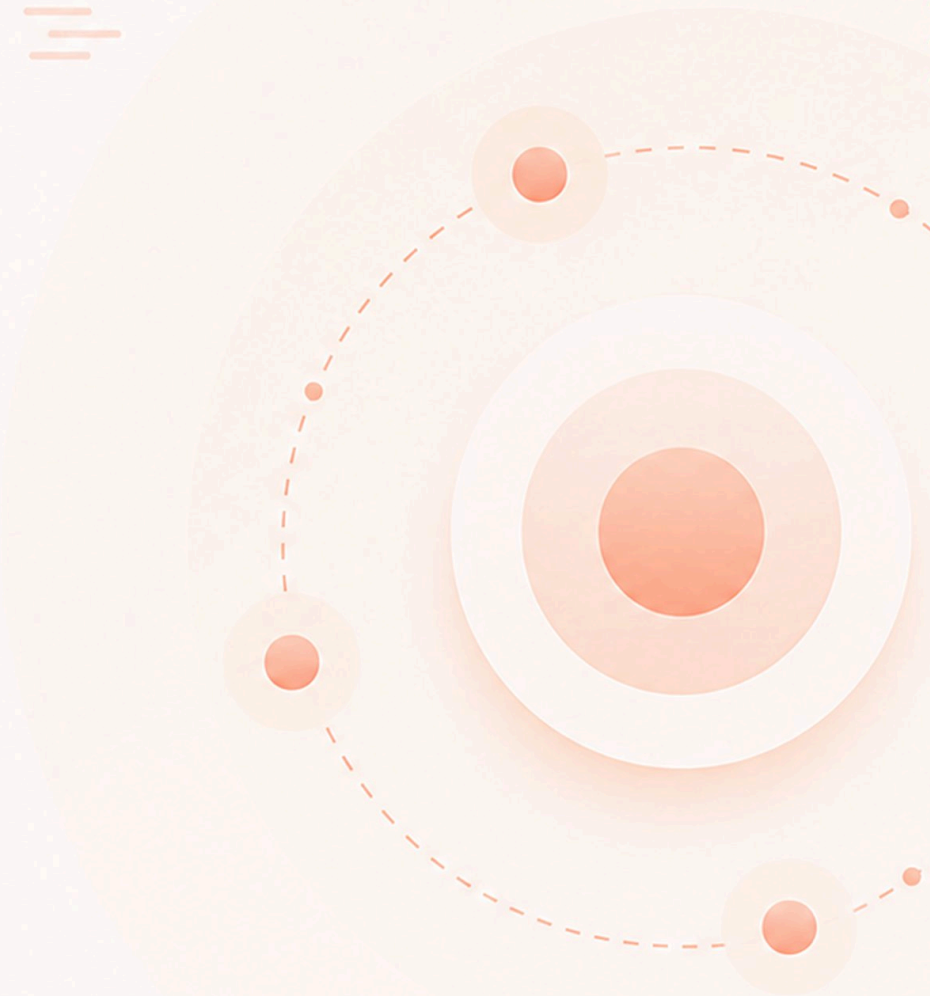
The GitHub interface for creating a new pull request titled fix/vault router security and pagination, showing the standard PR creation flow before the agent triggers



The pull request immediately after creation showing Some checks have not completed yet with the PR-Agent Auto Review check Started now, demonstrating the agent triggering automatically within seconds of the PR being opened

CHAPTER 4

Features & Workflows



4. Features and Workflows

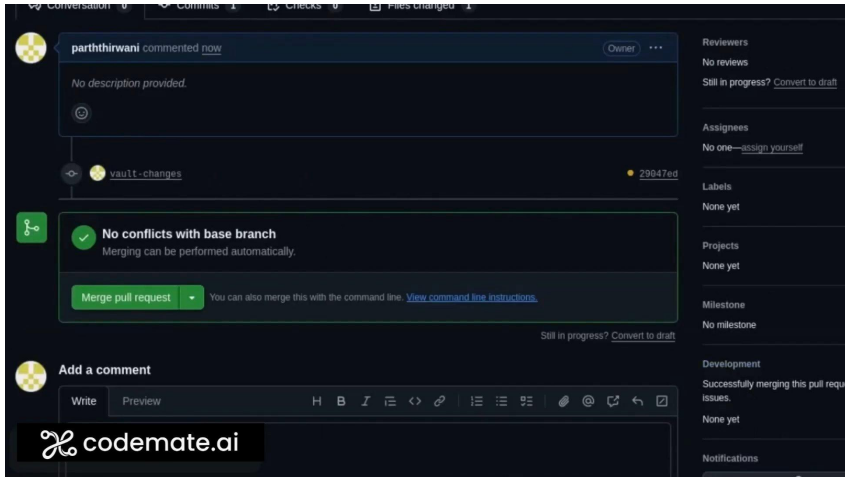
4.1 Automatic Review

The automatic review runs on every pull request without any action required. It includes a PR description covering type and summary, a File Walkthrough, a PR Reviewer Guide with key observations, an estimated review effort score from 1 to 5, and an automatically applied label.

When to use it: The automatic review is always running. Every PR opened in a repository where the agent is installed receives it within 45 to 60 seconds of being opened.

How it works: The agent receives a webhook event, fetches the PR diff, reads changed files, runs analysis, posts output as a comment, applies the effort label, and updates the check status to show completion.

> **Example**
>
> A developer opens PR #41 titled "fix/vault router security and pagination." Within seconds PR-Agent Auto Review appears as in progress. Within 45 seconds it completes, having posted a PR description identifying the change type, a four-point description, a File Walkthrough, a PR Reviewer Guide flagging two security concerns, and a Review effort 2/5 label applied automatically.



The pull request before the agent's output has posted, showing only the base GitHub checks: No conflicts with base branch and the Merge pull request action

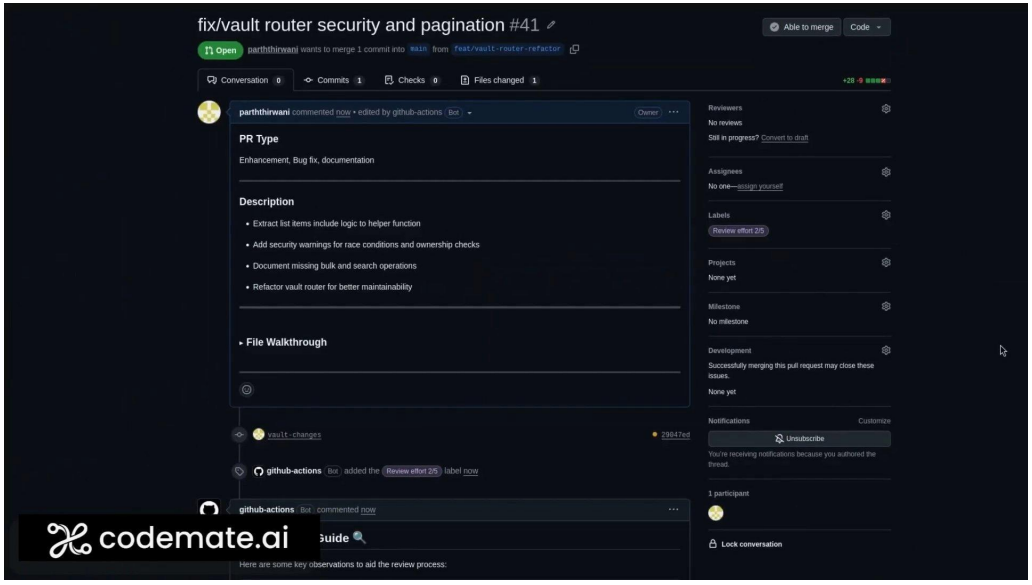
4.2 /describe

The /describe command generates or regenerates a structured description of the pull request, including PR type, a plain-language description as a bullet list, and a File Walkthrough table listing every changed file with a description of what changed.

When to use it: Use when the automatic description needs to be regenerated after significant new commits, when the PR has been substantially revised, or when the developer wants to use the generated description as the PR's official description.

How to use it: Type /describe in the Add a comment box and click Comment. The agent reads the current PR diff and posts a new structured description within 30 to 60 seconds.

> **Example**
>
> A developer opens a PR that refactors the vault router with security improvements. The agent automatically generates a description identifying it as Enhancement, Bug fix, and documentation type with a four-point description. The developer copies the generated description into the PR's official description field without writing anything themselves.



The /describe output showing PR Type as Enhancement, Bug fix, documentation, a Description with four bullet points covering the key changes made in the PR, and the Review effort 2/5 label applied

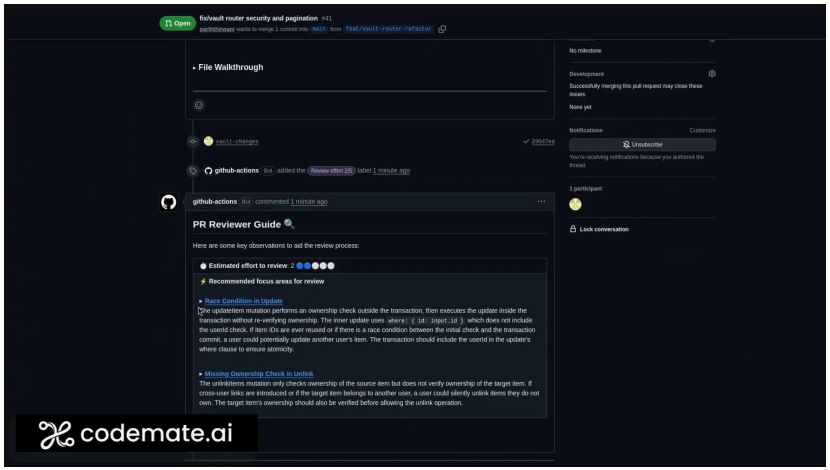
4.3 /review

The /review command triggers a full AI review of the PR diff, producing the PR Reviewer Guide with estimated review effort, recommended focus areas with expandable findings, and for each finding a detailed explanation, the specific code location, and guidance on what should change.

When to use it: Use when you want a fresh review after changes have been pushed following the initial automatic review, when the automatic review was not comprehensive enough for a particularly complex PR, or when confirming an issue has been addressed.

How to use it: Type /review in the Add a comment box and click Comment. The agent analyzes the current PR diff and posts a new PR Reviewer Guide within 30 to 60 seconds.

> **Example**
>
> A team runs /review on the vault router PR. The agent returns a Reviewer Guide estimating effort at 2 out of 5 and flagging two security concerns: a race condition where ownership is checked outside the transaction but the update runs inside without re-verifying, and a missing ownership check in the unlinkItems mutation. Both findings include the exact file location, a detailed explanation, and a recommendation.



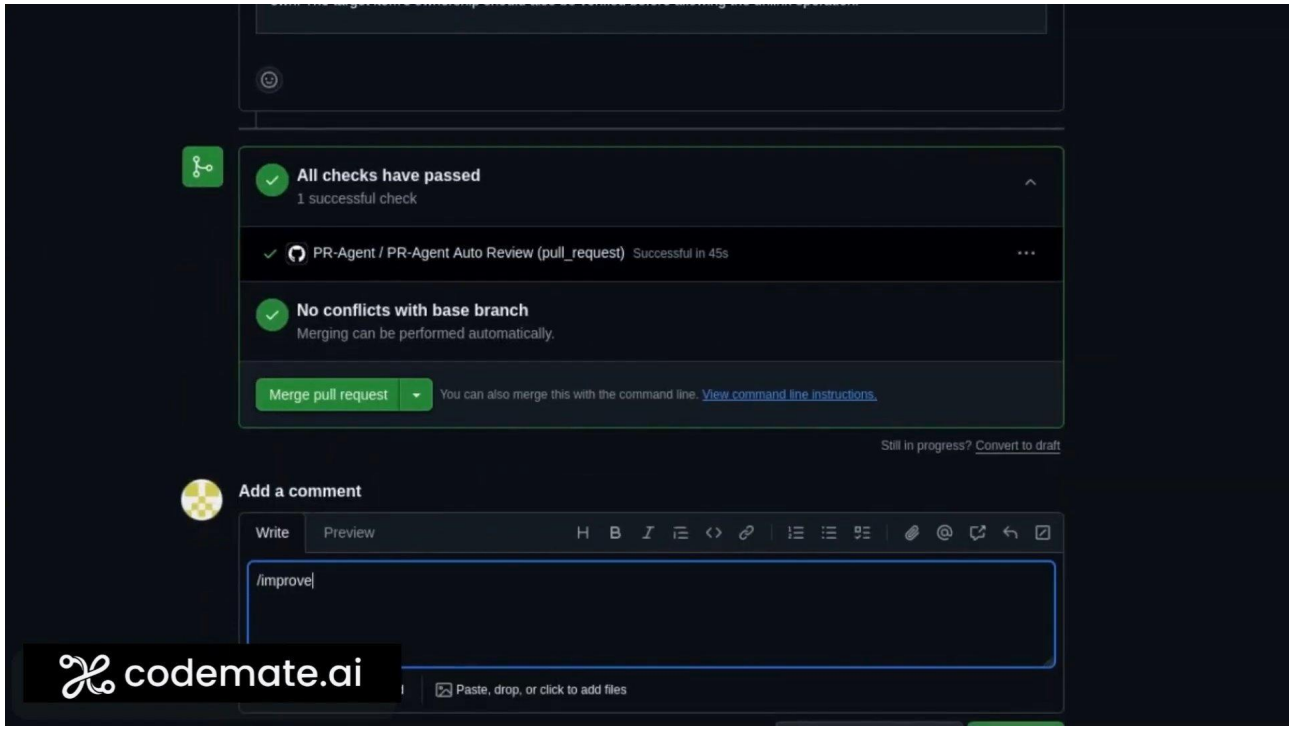
The PR Reviewer Guide showing Estimated effort to review at 2 out of 5, Recommended focus areas with Race Condition in Update and Missing Ownership Check in Unlink, each with detailed explanations of the security risk

4.4 /improve

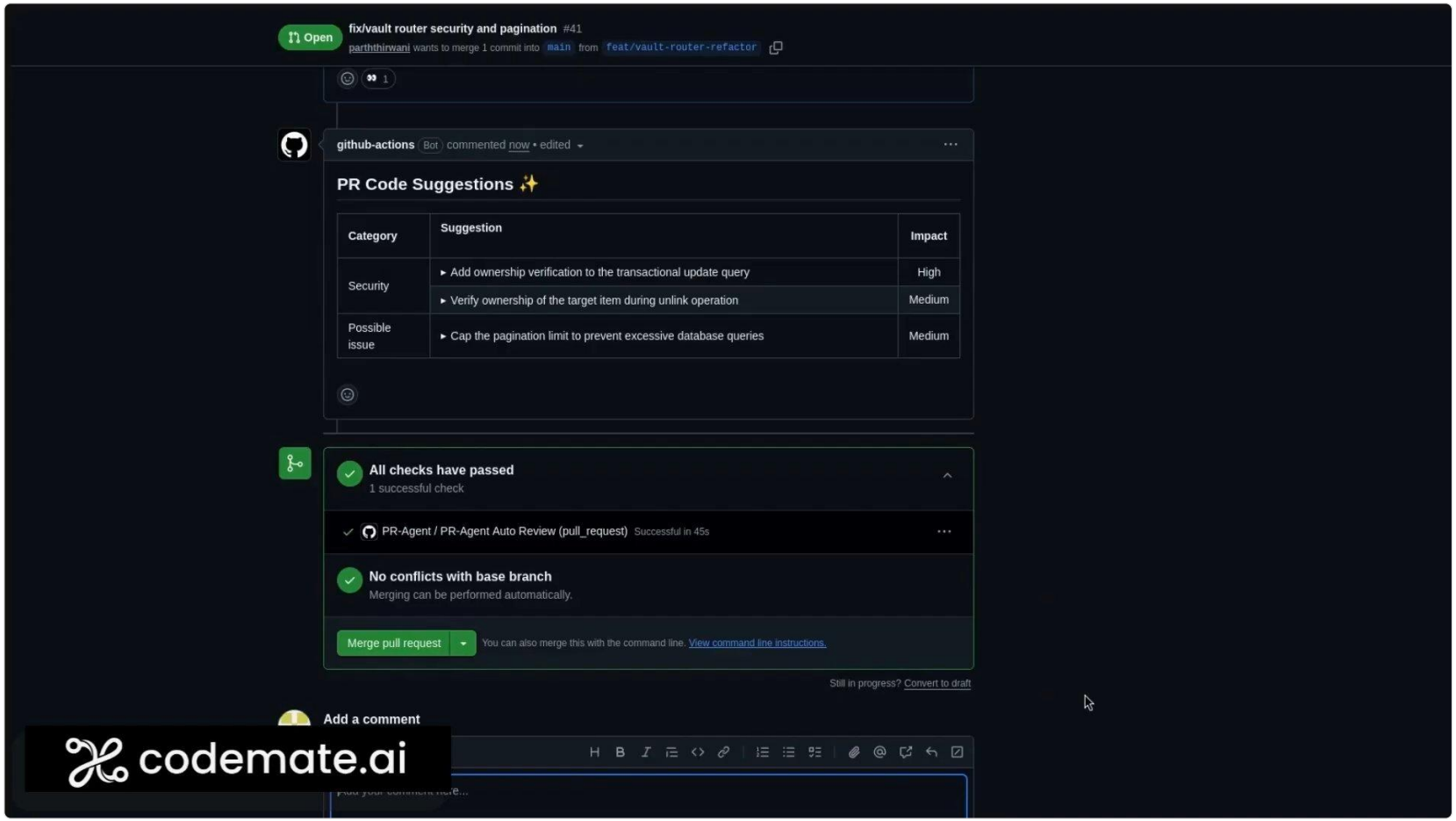
The /improve command generates ranked, categorized, actionable code improvement suggestions as a PR Code Suggestions table with Category, Suggestion description, and Impact level columns. Each suggestion is expandable to show a full explanation, the specific file and line range, a diff showing current code in red and suggested change in green, and a suggestion importance score from 1 to 10.

When to use it: Use when you want specific actionable code changes beyond the high-level observations in the reviewer guide, when the developer wants to see exactly what the code should look like after each fix, or when preparing a PR for merge and wanting a final pass of improvement suggestions.

How to use it: Type /improve in the Add a comment box and click Comment. The agent generates the suggestions table. Each row can be expanded by clicking the arrow to reveal the full explanation and diff.



The PR comment input box with /improve typed and ready to be submitted, with All checks have passed and PR-Agent Auto Review Successful in 45s confirming the initial automatic review has already completed



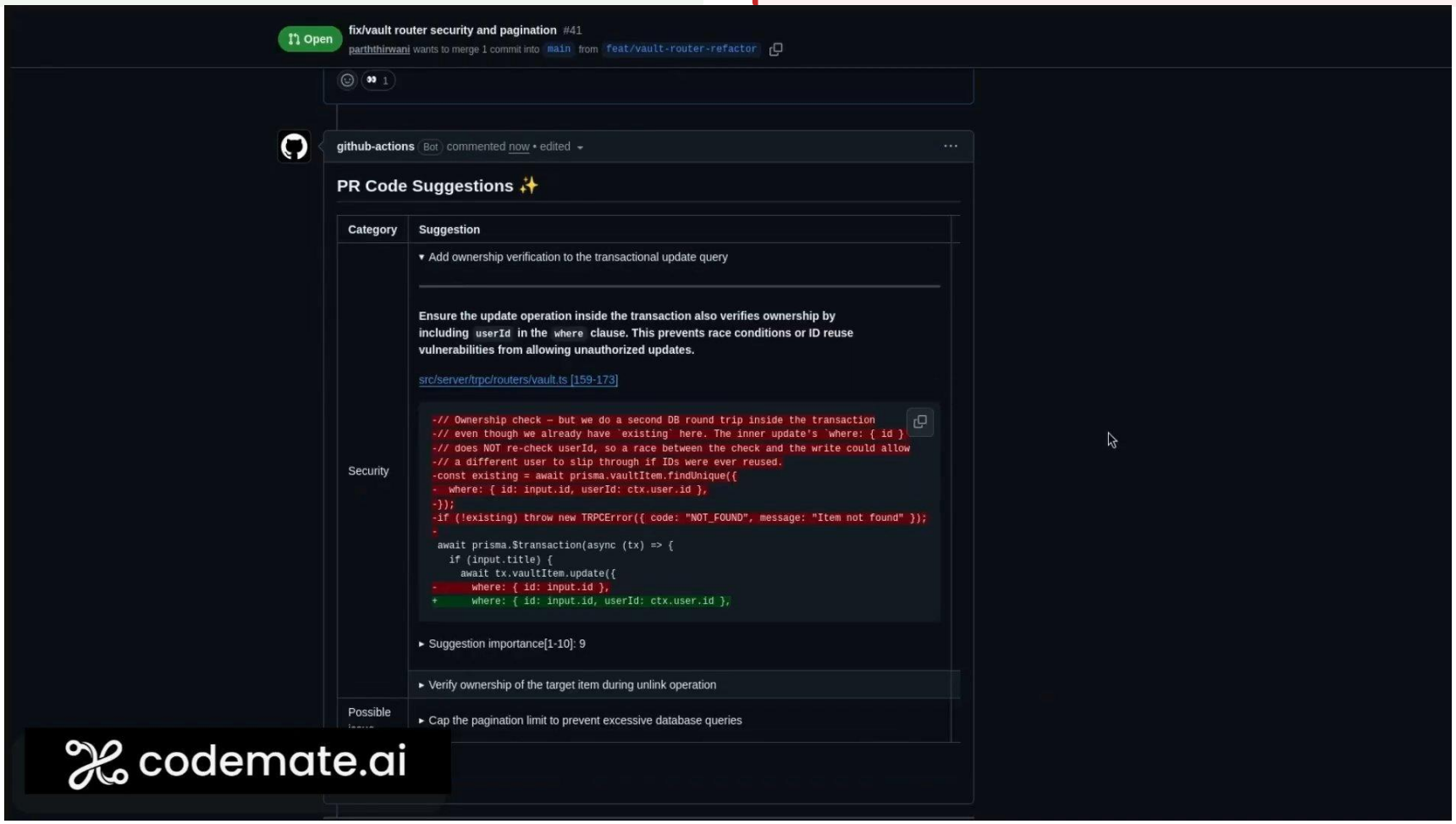
The completed /improve output showing PR Code Suggestions with three suggestions: Security category with Add ownership verification at High impact and Verify ownership of target item at Medium impact, and Possible issue with Cap the pagination limit at Medium impact

GOOD: Run /improve before merge

The developer runs /improve as a final pass, applies the High-impact security suggestion directly from the generated diff, and merges with the ownership check in place.

RISKY: Skipping the final pass

The developer merges right after the automatic review without running /improve, missing the concrete ownership-verification fix that the suggestion table would have surfaced with a ready-to-apply diff.



The expanded first suggestion showing the full detail for adding ownership verification, with a plain-language explanation, the file reference, a diff showing the vulnerable code in red and the corrected code in green with userId added to the where clause, and a Suggestion importance score of 9 out of 10

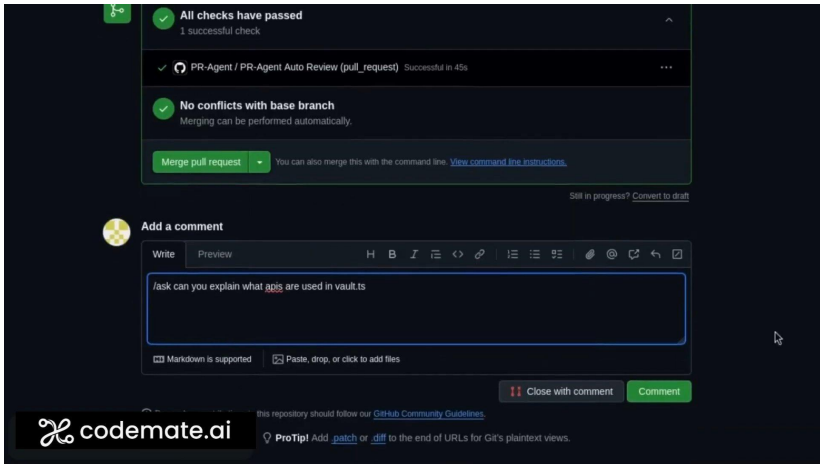
4.5 /ask

The /ask command lets any team member ask the PR Review Agent a free-text question about the pull request. The agent reads the question, the PR diff, and the changed files, and responds with a detailed, accurate answer posted as a comment with the question echoed back followed by the answer.

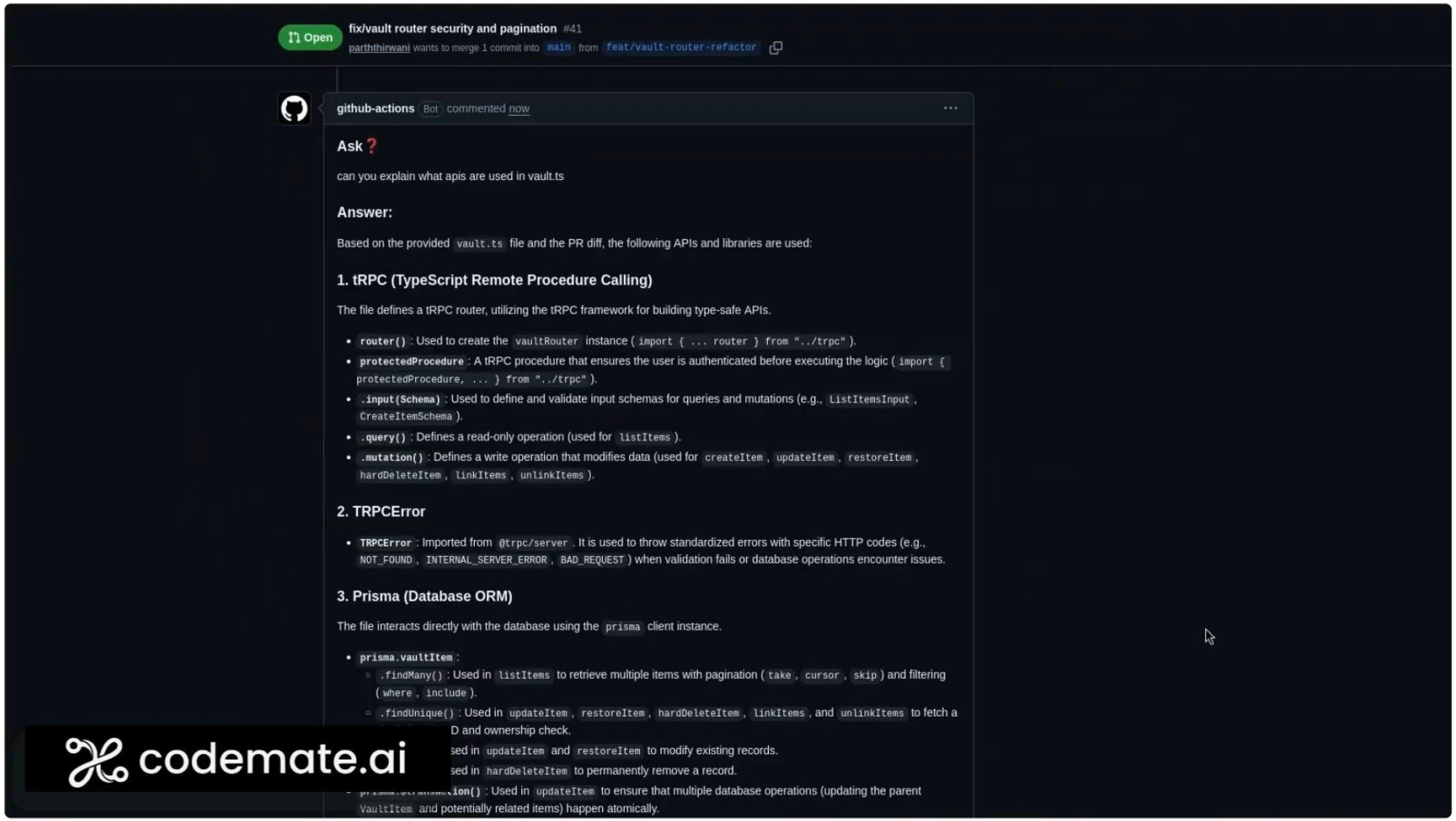
When to use it: Use when a reviewer has a specific question about the changed code, when a non-engineer needs to understand what a technical change does, when an engineer is unfamiliar with a specific API used in the PR, or when a reviewer wants to understand the implications of a specific approach before approving.

How to use it: Type /ask followed by your question in the Add a comment box and press Comment. For questions about specific files, include the filename in the question to help the agent focus its response.

> **Example**
>
> A new engineer reviewing a PR that touches vault.ts types "/ask can you explain what apis are used in vault.ts." The agent responds with a structured explanation covering tRPC with all methods explained, TRPCError for standardized error handling, and Prisma with every method used in the file documented with its specific purpose. The engineer can now review the PR meaningfully without opening the codebase or asking a colleague.



The /ask command fully typed as "/ask can you explain what apis are used in vault.ts" in the PR comment box, ready to be submitted



The PR Review Agent's complete /ask response showing Ask with the question echoed back, followed by an Answer covering tRPC explaining all methods, TRPCError for standardized error handling, and Prisma as the database ORM with all methods documented with their specific uses

4.6 /update_changelog

The /update_changelog command generates a changelog entry for the changes introduced in the PR and either creates or updates the CHANGELOG.md file in the repository, following semantic versioning conventions.

When to use it: Use for any PR introducing a significant change, including new features, bug fixes, security patches, and breaking changes. Running this at merge time rather than at release time keeps the changelog current automatically.

How to use it: Type /update_changelog in the Add a comment box and click Comment. The agent reads the PR and generates a structured changelog entry formatted to match the existing CHANGELOG.md style.

> **Example**
>
> A team merges a PR adding ownership verification to database mutations and capping the pagination limit. They run /update_changelog before merging. The agent generates an entry with a Security section documenting the ownership verification fix and a Changed section documenting the pagination limit cap, formatted consistently with the existing changelog.

4.7 /add_docs

The /add_docs command generates inline documentation for functions, methods, and modules changed in the pull request. For TypeScript it generates JSDoc comments; for Python it generates docstrings. The output is posted as a PR comment showing exactly where each comment should be added and what it should say.

When to use it: Use when a PR introduces new functions that lack documentation, when the PR touches a public API or shared utility, or when a reviewer requests documentation before approving.

How to use it: Type /add_docs in the Add a comment box and click Comment. The agent reads the changed files and generates documentation for every function or method that was added or modified.

> **Example**
>
> A PR adds three new tRPC mutations to the vault router. A reviewer runs /add_docs. The agent generates JSDoc comments for each mutation explaining what it does, what input schema it validates against, what it returns, what TRPCError codes it can throw and under what conditions, and any side effects on related data.

4.8 /generate_labels

The /generate_labels command analyzes the pull request and applies labels based on its content, covering type of changes, affected codebase areas, estimated complexity, and special characteristics such as security relevance or breaking changes. If needed labels do not exist, the agent creates them.

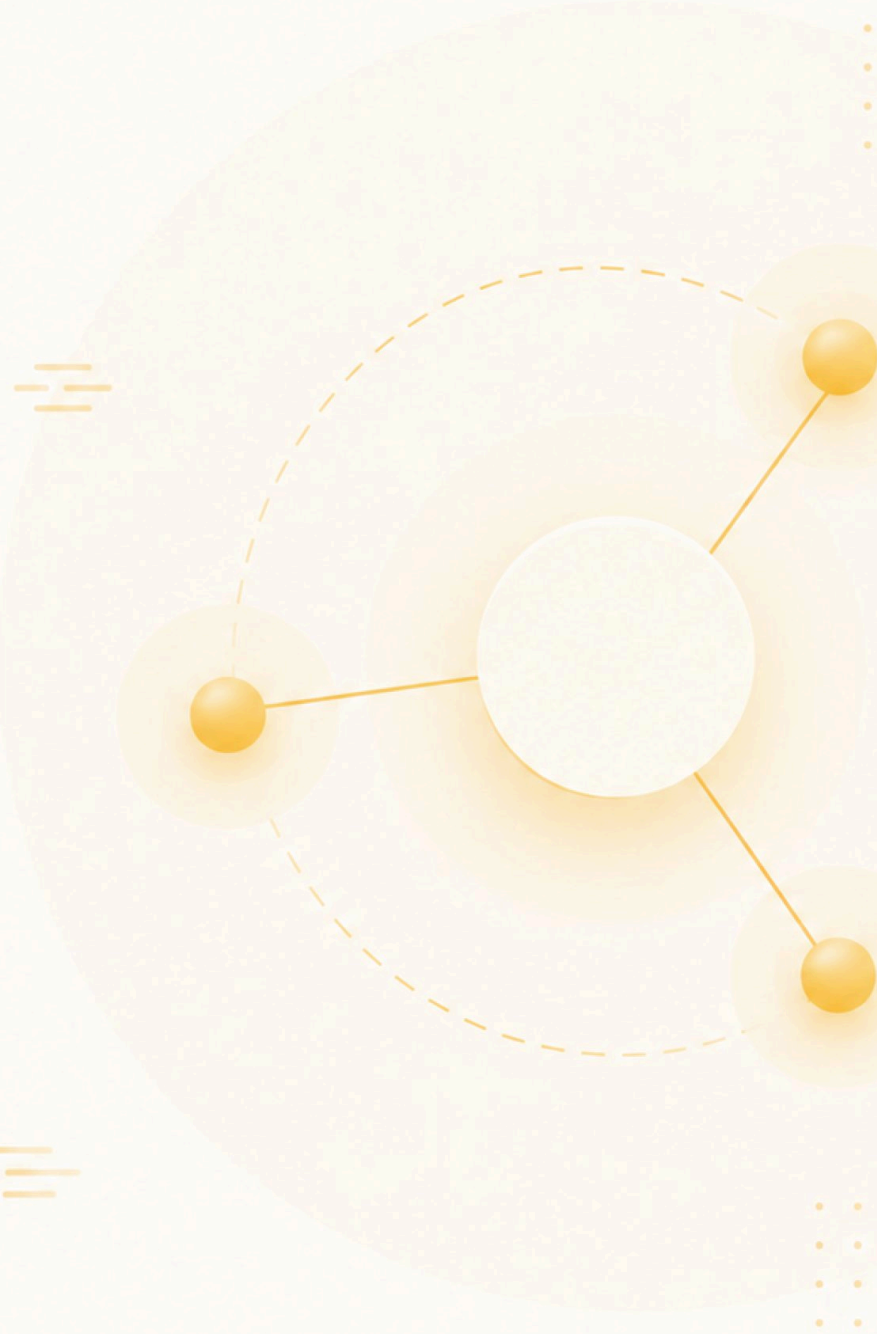
When to use it: Use when the PR has not been labeled by the developer, when consistent labeling is needed for filtering and reporting, or when establishing a consistent labeling practice on a new repository.

How to use it: Type /generate_labels in the Add a comment box and click Comment. The agent analyzes the PR and applies labels automatically. They appear in the PR sidebar immediately after processing.

> **Example**
>
> A team opens a PR touching authentication, adding security fixes, and including documentation updates. They run /generate_labels. The agent analyzes the changes and applies security, bug fix, documentation, and enhancement labels based on the actual content, making the PR immediately filterable alongside all other security-related PRs in the repository list.

CHAPTER 5

Use Cases



5. Use Cases

Maintaining code quality at scale

Trigger: An engineering team has grown to the point where consistent code quality is difficult to enforce across every open PR.

- The automatic review applies the same thorough first pass to every PR regardless of how many are open at once
- Issues a tired or rushed human reviewer might miss are caught every time

Result: A consistent quality bar across the codebase that does not degrade as PR volume grows.

Catching security vulnerabilities before merge

Trigger: A team wants security issues caught while they are cheapest to fix, before code is merged rather than after it reaches production.

- Every review specifically checks for race conditions, missing ownership checks, input validation gaps, SQL injection risks, XSS vulnerabilities, and insecure token storage
- Run /review for a focused pass, or /improve for a ready-to-apply diff fixing the issue

Result: Vulnerabilities identified and fixed before they ever reach production, with the exact file location and recommended change.

Reducing review bottlenecks for senior engineers

Trigger: Senior engineers are spending too much time on routine reviews and cannot easily tell which PRs need dedicated focus.

- The review effort score tells a reviewer at a glance whether a PR needs minutes or dedicated time
- A score of 2 out of 5 signals a quick pass; a 4 or 5 signals the reviewer should block time

Result: Reviewer attention is allocated where it matters most instead of being spread evenly across every PR.

Generating PR documentation automatically

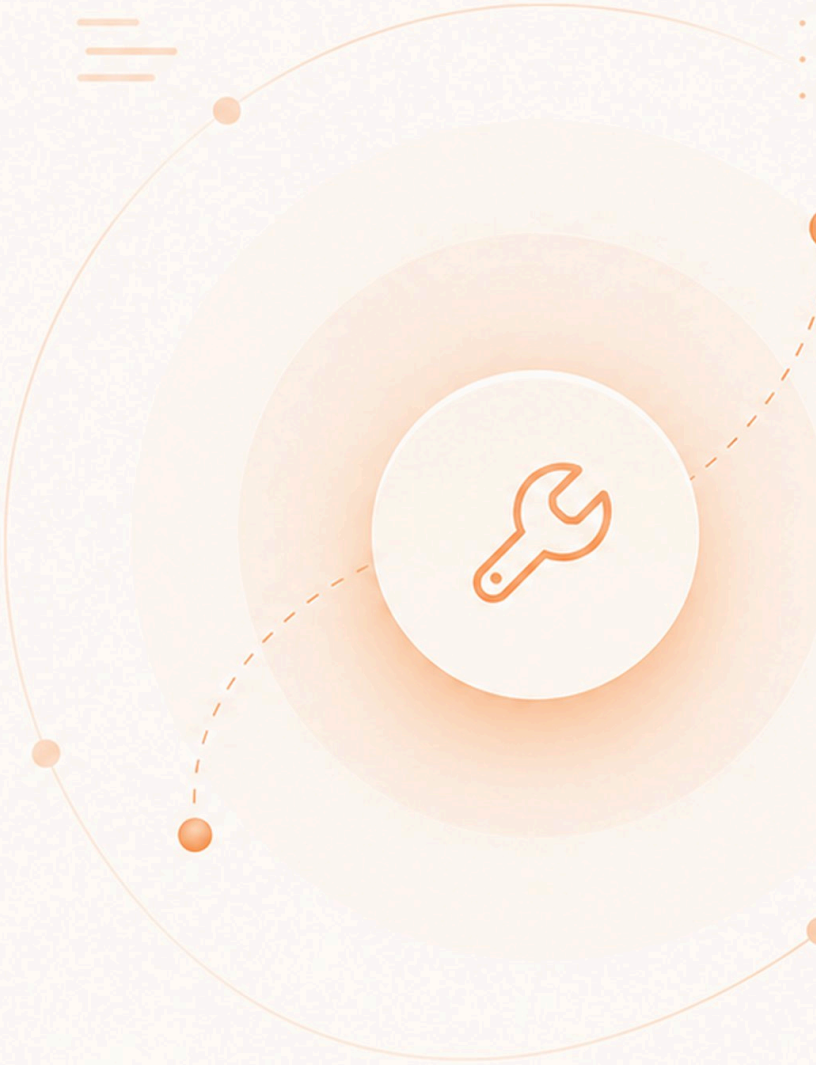
Trigger: Developers routinely open PRs with no description, leaving reviewers to reconstruct context from the diff alone.

- The automatic review, or /describe on demand, generates a structured description covering type, summary, and a file-by-file walkthrough
- The generated description can be copied directly into the PR's official description field

Result: Every PR has a clear, structured description without any extra effort from the developer who opened it.

CHAPTER 6

Tips, Patterns, & Troubleshotting



6. Tips, Patterns and Troubleshooting

Tips and patterns

- Let the automatic review run before typing any command; it already covers the description, reviewer guide, effort score, and label, so a manual command is only needed for something the automatic pass does not produce.
- Re-run /review after pushing new commits rather than relying on the original automatic review, since the reviewer guide reflects the diff at the time it was generated.
- Use /improve as a final pass before merge: its suggestions come with ready-to-apply diffs, not just observations.
- Review comments from /describe and /review are updated in place on re-run, so the PR thread stays clean instead of accumulating duplicate comments.
- Include the filename in an /ask question to help the agent focus its answer on the right part of the diff.

Troubleshooting

Issue	Cause	Solution
The agent is not posting comments on pull requests	The version control integration or webhook is inactive, or organization access is not configured.	Confirm the PR Review Agent is correctly integrated with your version control platform (GitHub, GitLab, Bitbucket, or Azure DevOps) and that the webhook or integration is active in your repository settings. Since this is an Enterprise product, verify your organization's access is configured correctly. If the issue persists, contact contact@codemate.ai for Enterprise support.
A manual command is not producing any response	The command was mistyped, or was not posted as a standalone comment.	Check the command matches exactly: /describe, /review, /improve, /ask, /update_changelog, /add_docs, or /generate_labels. Post it as its own comment in the Add a comment box rather than appending it to other text, and allow up to 60 seconds for the agent to respond.

Issue	Cause	Solution
The PR thread is cluttered with multiple review comments	/describe or /review was re-run expecting a new comment each time.	This is expected in the other direction: both commands update their existing comment in place rather than posting a new one. If duplicate comments are appearing, confirm you are not running an older self-hosted version alongside the current integration.
The review effort label seems inconsistent with the size of the PR	The effort score reflects review complexity, not line count.	A large PR that only touches generated files or simple boilerplate can score low, while a small PR touching authentication or payment logic can score high. Treat the score as a complexity and risk signal from /review, not a proxy for diff size.
The team needs the agent to run inside its own infrastructure	Data residency, compliance, or air-gapped requirements prevent using the hosted SaaS integration.	CodeMate offers a fully self-hosted deployment option for the PR Review Agent, allowing it to run entirely within your own infrastructure. Contact contact@codemate.ai to discuss self-hosted deployment and Enterprise access.

CHAPTER 7

Resources and Reference

7. Resources and Reference

Where to go next

- Product documentation: docs.codemate.ai
- CodeMate website: codemate.ai
- Enterprise access and self-hosted deployment: contact@codemate.ai
- For issues beyond this guide, see the CodeMate PR Review Agent FAQ and Troubleshooting documents.

Quick checklist

Use this before considering your PR Review Agent setup complete.

- Confirmed Enterprise access and installed the GitHub App (or GitLab, Bitbucket, Azure DevOps equivalent) on the right repositories
- Verified the automatic review triggers within a minute of opening a test pull request
- Comfortable using all seven manual commands: /describe, /review, /improve, /ask, /update_changelog, /add_docs, and /generate_labels
- Decided whether a self-hosted deployment is needed for compliance or data residency requirements
- Adopted /improve as part of the pre-merge checklist for a final pass on code suggestions