ATLASSIAN

AI-powered code reviews: a playbook for engineering leaders

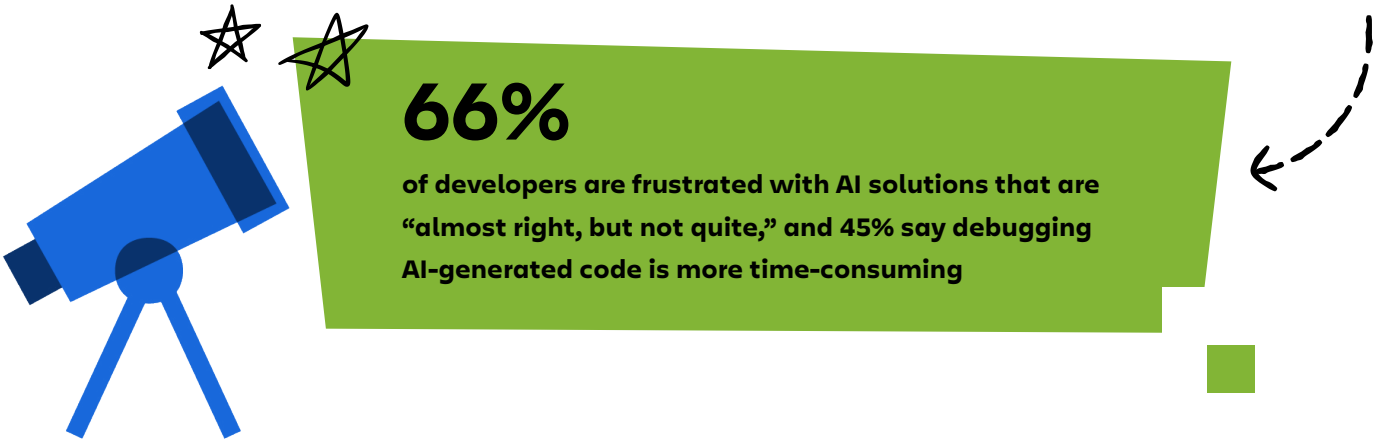
Learn how Rovo Dev can help you shorten PR cycle times and boost code quality at scale.

Table of contents

Code review is becoming a bottleneck

The amount of code flowing through your pipelines has never been higher. According to the 2025 Stack Overflow Developer Survey, 84% of developers are using or planning to use AI tools in their workflows. The upside is obvious: developers are shipping much, much faster. But the downside is becoming just as clear: more subtle bugs, more hidden risk, more inconsistent adherence to standards, and a mounting layer of tech debt created at machine speed.

Developers are already feeling this tension. The same survey shows that 66% of developers are frustrated with AI solutions that are “almost right, but not quite,” and 45% say debugging AI-generated code is more time-consuming.



Traditional, human-only code review hasn't scaled to match this new reality. AI coding assistants have dramatically increased the volume and velocity of code changes, while review capacity and governance have stayed roughly the same. Pull requests are opening faster than your team can reasonably keep up. Queues pile up. Reviews get rushed because reviewers rarely have the time or full context to deeply understand complex, AI-assisted changes.

Based on data from our own engineering teams, we found that pull requests wait an average of 18 hours before they receive a first comment. Reviewers are stretched thin, context is fragmented across teams and services, and important issues were slipping through.

The challenge: AI is speeding up code creation, but code review is still limited by human attention and time. More AI-generated code without a corresponding change in how you review and govern it simply means more bugs, more risk, and more rework pushed downstream.

In this guide, we'll show you how AI-powered code review with Rovo Dev can help you turn code review from a human checkpoint into a scalable system that improves code quality, shortens review cycles, and reduces risk for your organization.

At Atlassian, teams using AI code review have reduced PR cycle times by **45%** while also cutting production incidents. For an enterprise with hundreds of developers, that translates into dozens more features shipped per quarter and fewer customer-impacting regressions.

Software complexity is exploding

The bottleneck you feel in code review isn't just about more PRs. It's about the kind of systems those PRs are touching.

Over the last decade, most architectures have shifted toward distributed services, feature flags, and continuous delivery. Teams are shipping more changes across more services, often with the same or fewer engineers. At the same time, AI has dramatically reduced the friction to generate code—developers can produce entire functions or files in seconds. The net effect is simple: **far more change, in far more complex systems, landing on the same limited pool of reviewers.**

Why this matters for code review

- **Human reviews don't scale with this reality:** Your senior engineers are reviewing dozens of PRs a week on top of incidents, design work, and mentoring. Attention is spread thin; important issues slip through.
- **Most risks are now systemic:** The hardest bugs live in interactions between services, performance under load, and subtle security and compliance edge cases, exactly the problems that are hardest to spot in a quick PR skim.
- **"Shift left" is still mostly aspirational.** You may have policies for security, compliance, and reliability, but enforcement often happens late in the pipeline or after an incident, not when the change is first proposed.

The role AI must play

AI has to move closer to where risk is introduced into the system:

- **Sit where risk enters the system: the PR**. AI should run on every pull request, surfacing likely risks, missing requirements, violations of coding standards, potential bugs, and maintainability issues while the change is still cheap to fix.
- **Understand your organization, not just your code.** That means knowing acceptance criteria in Jira, project context from Confluence, your coding standards, security and compliance rules, and team conventions, so it can enforce the way your organization builds software, not just generic best practices.

The rest of this guide walks through how you can design a review system that combines always-available AI review with human judgment, so quality and governance can keep pace with the speed and complexity of modern, AI-accelerated development.

5-step playbook to scale code review

Modern code review should be designed as a scalable system, not just a developer ritual. Start by defining what "good" looks like for your organization in terms of your standards and risk thresholds, and then make AI review a default, always-on guardrail for every PR.

1. Define success metrics

Align your teams on why you're doing this and how you'll judge success.

Typical goals	Decide what you'll measure
▪ Shorten PR cycle time (open → merge)	▪ PR cycle time
▪ Reduce post-release bugs/incidents	▪ Incident / rollback rates
▪ Free up senior engineer time from repetitive review work	▪ Reviewer load (hours/week and # reviews per engineer)

2. Start with 1–2 pilot teams

Treat this as a controlled experiment, not an org-wide rollout.

Choose teams with:	Limit scope:
▪ Clear ownership and stable services	▪ A defined set of repos
▪ Tech leads willing to experiment and give feedback	▪ A clear pilot window (e.g., 4–8 weeks)

Baseline the metrics above before turning AI review on.

3. Connect your AI tools to your organization's coding standards

Connect your AI tools to your organization's coding standards so they enforce your rules, not just generic best practices.

Centralize expectations for style, testing, security, and compliance in a single source of truth, then configure your AI tool to reference and encode those rules, such as required auth patterns, logging, or data-handling constraints. This keeps AI feedback specific, consistent, and aligned with how your teams actually build and ship software.

4. Keep humans as final approvers; AI as a baseline reviewer

Position AI as the baseline reviewer that runs on every PR and flags obvious bugs, missing requirements, style violations, and risky patterns, while humans remain the final approvers.

Require at least one human approval on all PRs, and have reviewers focus their time on design decisions, risks, and trade-offs instead of routine checks.

5. Tune, govern, and scale

Treat this as an evolving system. Use developer feedback and comment dismissal patterns to update your coding standards to dial back noisy checks and strengthen the ones that reliably catch real issues. The goal is to train AI to provide comments that are meaningful so developers trust AI feedback, rather than tune it out.

Assign clear ownership and cadence for maintenance of these coding standards (e.g., platform leads updating it quarterly). Define how exceptions are requested and approved, and ensure they're documented. This gives leaders and developers confidence there is a system in place to manage the quality of AI output.

Once pilots show improved metrics and acceptable noise levels, you can onboard additional teams using the same pattern: baseline → pilot → tune → scale.

Benefits of this new system

When AI is wired into your review process this way, you'll deliver consistent outcomes without burning out your best engineers.

- **Consistent, predictable quality across teams.** Every PR gets checked against the same standards, not just the preferences of whoever happens to be on review. As teams adopt different stacks and patterns, you still get a common baseline for style, security, and architectural rules.

- **Faster, richer feedback loops.** Developers don't wait days for a drive-by review. AI flags issues on every PR, with concrete suggestions and links back to your standards and Jira context. That tight loop improves both code quality and provides valuable feedback to your newer engineers.

- **Higher code quality with lower risk.** Because AI reviews every change in context, not just syntax, you catch more issues before they ship: missing requirements, unsafe patterns, drift from acceptance criteria, and violations of your security or compliance rules. Incidents drop, and your overall risk profile improves over time.

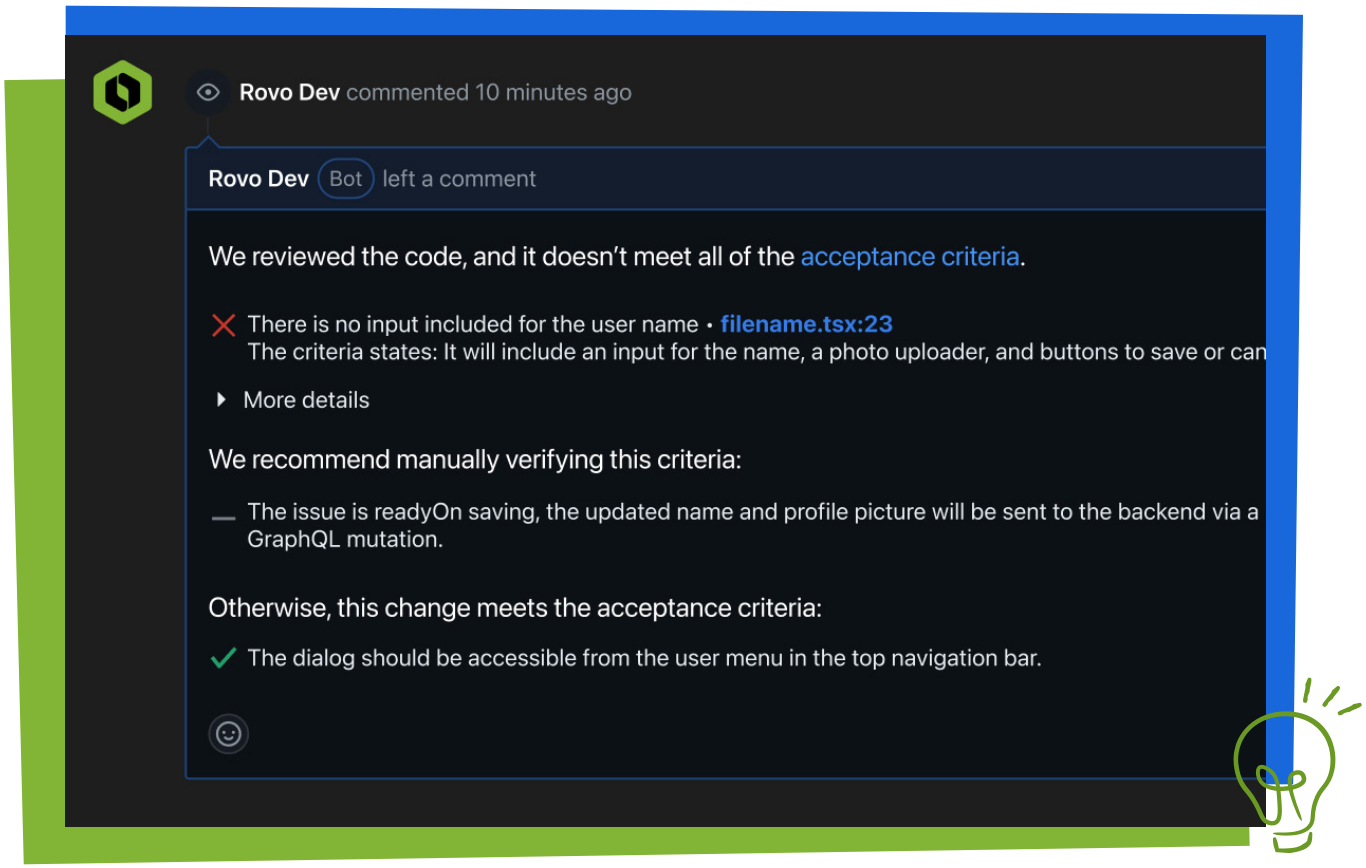
Once you design review processes in this way, AI is no longer just a tool. It starts to behave like a tireless senior engineer embedded in every team: enforcing your standards, watching for risk, and freeing your actual senior engineers to focus on the hard problems only they can solve.

AI-powered code review with Rovo Dev

Rovo Dev is a contextual AI agent (for Bitbucket and GitHub) built to help professional software engineering teams accelerate software development by assisting with planning, code generation, reviews, and automating repetitive work at scale.

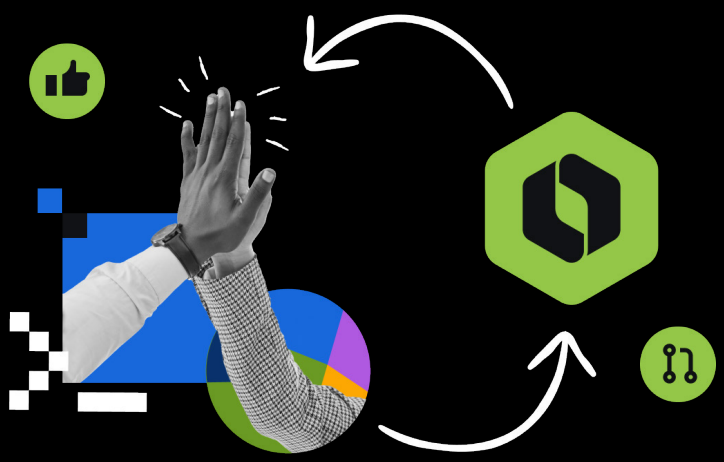
What can Rovo Dev do for code review?

Rovo Dev's capabilities include an AI-powered code review agent that runs automatically on your pull requests, checks code against your organizations coding standards and whether it meets the acceptance criteria for the work as listed in the Jira ticket. Based on findings, it posts clear, actionable suggestions on issues like readability, bugs, and maintainability, and gives authors the option to apply simple fixes in-line with a single click.



How it works

- **Understands your organization's context.** Rovo Dev ingests context from relevant files, your codebase, and your org's custom coding standards e.g., you can write a custom coding standards document detailing your:
  - lint rules
  - design patterns
  - security & compliance policies
  - API contracts
  - + anything you define in your coding standards file (which serves as model context).
- **Knows acceptance criteria from Jira.** When your PR is linked to a Jira issue, Rovo Dev automatically reviews code changes against the acceptance criteria listed in the Jira ticket.
- **Reviews every PR.** Rovo Dev runs on every open pull request, parses the diff, and analyzes code changes.
- **Leaves comments with suggestions:** Each comment explains why the issue matters and, where possible, includes a concrete code suggestion. Developers can apply simple suggestions in one click.
- **Native to your PR flow.** Comments appear inline in GitHub and Bitbucket in your PR view so developers stay in their flow and review comments from AI and humans together.



Atlassian teams shortened PR cycle times by 45%

In under a year, internal teams at Atlassian reduced average PR cycle time by 45% by rolling out Rovo Dev as an AI-powered code reviewer across thousands of engineers.

Rovo Dev now delivers instant feedback on every PR, flags bugs and policy gaps, and aligns changes to Jira acceptance criteria, cutting first review wait times from ~18 hours to near-zero. New engineers who used Rovo Dev merged their first PR five days faster, improving both throughput and onboarding.

"We didn't just build a tool – we built a whole new way of working. Now, we're seeing teams across Atlassian moving faster, staying aligned, and innovating continuously. Imagine what your team can do when everyone's working from the same playbook."

RAJEEV RAJAN
Chief Technology Officer at Atlassian

Human vs. AI code review

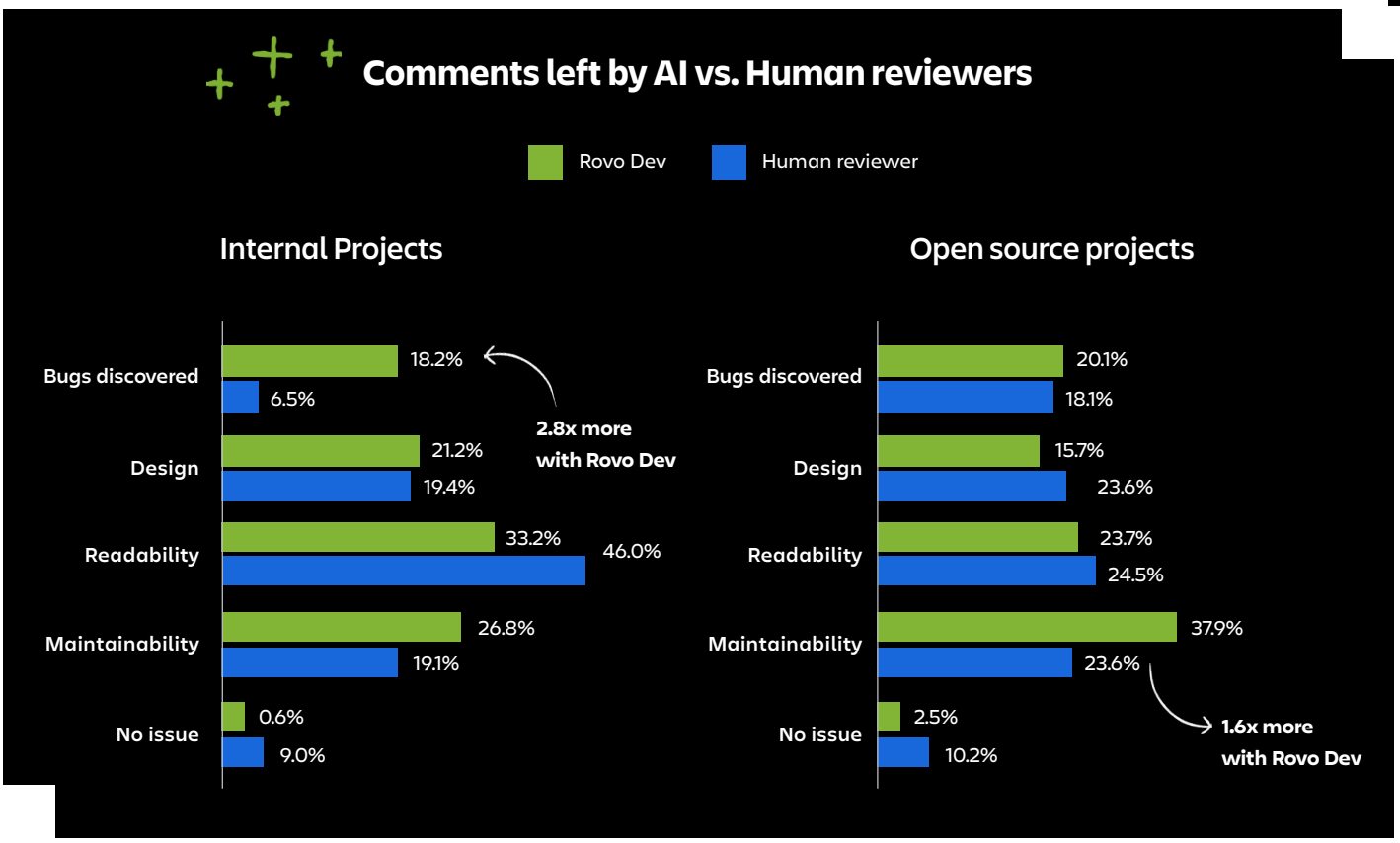
To compare the types of comments from Rovo Dev and human reviewers, we used OpenAI's GPT-4.1 to automatically classify thousands of code review comments into four types: readability, bug, maintainability, or design.

We analyzed around 4,000 review comments from Atlassian internal projects and 1,000 comments from open source projects. The key distinction is that internal projects mainly use Typescript, Javascript, and Kotlin, while open source projects are mostly Java, C++, and Python.

Results:

When we compared Rovo Dev's code review comments to those written by humans, we found each brings unique strengths, and complement each other.

Rovo Dev generated a higher proportion of bug (2.8x) and maintainability (1.6x) comments, while human reviewers focused more on readability and slightly more design. In addition, Rovo Dev is focusing only on actionable comments with only 0.6% classified as "No Issue". This means AI is spotting bugs and maintainability issues at scale, so your senior engineers don't have to spend their limited time on nitpicks and mechanical checks. Instead, they can focus their reviews on what humans do best: system design, trade-offs, and leaving feedback to mentor newer developers.



Our customers love it

"Rovo Dev Code Reviewer is an incredible sense check on our pull requests. With no additional effort, every PR is analyzed to find those issues that are out of reach of a linter like variable naming, missing comments on complex code, loose typings and common gotchas with standard protocols. It has earned its place in our workflow as a complement to traditional linting and testing."

Adam Ahmed
CTO at released.so

"Rovo Dev became an invaluable tool for our engineers to do reliable and accurate code reviews, saving them dozens of hours of manual reviews. Especially the automated acceptance tests that ensure code changes align with the description in Jira are a game changer for us."

Alexander Grzesik
Chief Architect, NEWWORK Software

"Rovo Dev code review strikes the right balance - it surfaces the real issues in a pull request without drowning developers in noise. Having AI working alongside our developers in this way makes reviews sharper, faster, more consistent and more useful across the team."

Jonathan Fishbein
VP R&D, OceanMD

"Rovo Code Reviewer has streamlined our PR workflow for the Zygos app. It handles translations, grammar corrections, and fixes–from minor bugs to complex issues. Developers spend far less time per review, even across hundreds of PRs each month. Quality improves while cycle time drops."

Marios Vasileiou
CTO at Softline Computer Systems LTD

Rovo Dev Implementation Guide

Here is a step by step process on how to set up your organization with Rovo Dev.

- Start by signing up for a free 30 day trial of Rovo Dev.
- If you haven't already done so, connect your source code management tool to Jira.
  - If you use Bitbucket Cloud, you can do so via the Bitbucket UI. Learn more
  - For GitHub, install and configure the GitHub for Jira app. Learn more
  - Make sure you use your Jira issue key in your branches and pull requests so AI gets context on the change and its acceptance criteria. Learn how
- **Enable AI code review in Bitbucket or GitHub.**
  - For Bitbucket: turn on Rovo Dev for the workspace and enable AI code reviews on each target repository.
  - For GitHub: enable code reviews at the organization level, then per repository.

  Learn more
- **Set up your coding standards.**
  - Create or refine a markdown document with your coding standards.
  - Configure Rovo Dev to use those standards as custom instructions for code reviews, and, where needed, add repo-level rules.

  Learn more
- **Iterate.**
- Use a handful of real pull requests from the pilot teams to validate integration, comment quality, and noise levels, then adjust your markdown rules as needed.

A simple ROI framework

Here is an ROI framework that you can use to estimate the impact of AI-powered code review in your organization.

Capture your current baseline

Before you start the pilot, ask teams to record:

- Avg PR cycle time (open → merge) Post-release bugs / incidents per sprint or month
- Reviewer load: reviewer hours/week on PRs (especially for senior engineers)

Most of these can be pulled from your existing DevOps tools (Bitbucket/GitHub + Jira, or your internal analytics). For reviewer load, you may have to get input from a subset of your engineers so you can estimate.

Estimate improvement by team

- Assume a conservative improvement, for example, 20–30% faster PR cycle time
- Apply a formula to estimate improvements across your team:

- **Hours saved per week** ≈ number of reviewers × current reviewer hours/week × expected % reduction
- **PR time saved per sprint** ≈ avg PR cycle time × expected % reduction × # PRs per sprint

Similarly, estimate the time saved with fewer incidents and rollbacks.

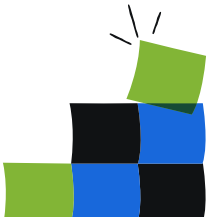
Finally, talk to your engineers and gather feedback: did the agent find bugs they might have missed? Did it suggest better methods? Do they feel it saves them time?

Track actual improvement

Make it easy to monitor. For the pilot repos,

- Put PR cycle time, and post-release bugs into a dashboard
- Ask a subset of developers to track their time spent on reviews

Review these numbers monthly/quarterly alongside lead time, deployment frequency, and incident metrics. This turns your AI code review pilot into a measurable, engineering-led investment.



What you're probably thinking

Is this worth the cost?

Running the pilot should improve code quality, shorten PR cycle time, reduce incidents and rollbacks, lighten senior engineers' review load, and speed new hire onboarding. When these metrics improve, the costs are justified.

Rovo Dev also gives you capabilities beyond code review - developers can generate code from Jira issue descriptions, use the IDE and CLI plugin for code generation, and use agentic CI/CD capabilities in Bitbucket to build workflows with natural language, and troubleshoot builds faster.

Is this just going to create noise?

You control the coding standards you set. Start with a focused, high-value ruleset and adjust it based on your teams feedback, reducing checks that get routinely ignored and strengthening those that catch real issues.

Will my developers push back?

How you present it matters. AI acts as a baseline reviewer handling repetitive checklist work, while humans remain final approvers on meaningful changes, focusing on design, architecture, and mentorship. Framed this way, and combined with managing your coding standards to remove noisy comments, developers will adopt it.

Will this disrupt how we ship?

Rovo Dev's code review lives in the PR view in Bitbucket or GitHub, with context from Jira and your coding standards, so engineers stay in their workflow and see rich, contextual feedback. AI comments appear like human comments, preserving the developer experience.

Is my code and data safe?

Rovo Dev operates within your existing stack - Jira, Bitbucket/GitHub, and Confluence - with the same security, compliance, and audit controls you already have. Rovo is designed with enterprise-grade privacy and security at its core. Learn more about Atlassian's privacy and security policies here.

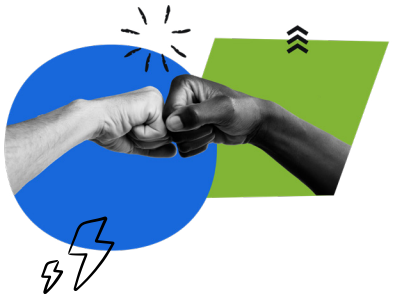
Conclusion

When you build a scalable process for code review, it stops being a bottleneck and becomes one of your strongest levers for code quality, speed and safety. This leads to:

- **Happier developers** spending less time on tedious checks and more time on design, problem-solving, and learning from each other.
- **Faster, more reliable deployments** as every change gets a consistent baseline review, and human reviewers focus on the highest-risk decisions.
- **Stronger governance** because standards are encoded once and enforced automatically, vs. being spread across silos.
- **Stronger technical leadership** as senior engineers spend less time gatekeeping individual PRs and more time mentoring developers.
- **A more resilient codebase** where security, compliance, and maintainability are baked into the daily flow of work.

AI-powered code review isn't just about speeding up PR cycle times. It's about building an engineering organization that can safely ship more software because quality, governance, and velocity scale together.

Resources:



ATLASSIAN Rovo Dev Sign up for your **free 30 day trial** of Rovo Dev.

Documentation

Demos:

- How to use PR Code Reviewer in Bitbucket Cloud | Rovo Dev | Atlassian
- Set up custom code review standards with Rovo Dev CLI | Rovo Dev | Atlassian
- Streamline your code reviews with automated acceptance criteria checks | Rovo Dev | Atlassian
- Automations with Rovo Dev | Rovo Dev | Atlassian

Explore our demo hub for the latest developer AI demos.