circleci

Is your CI/CD platform holding you back?

Get the complete guide to evaluating, scoring & switching platforms—with ready-to-use RFP template included.

(Re)Evaluating CI/CD: A guide for 2025 and beyond

THE DECISION TO RIP and replace a CI/CD platform isn't usually made on a whim. The conversation usually starts after enough friction builds up: slow pipelines, reliability issues, growing maintenance work, or developer complaints that don't go away.

Sometimes the problem is a legacy system held together with custom scripts and undocumented know-how. Other times, it's a newer tool that comes bundled with a suite of enterprise software—convenient at first, but never quite a perfect fit for how your teams actually build and ship software.

Either way, you reach a point where the cost of staying put outweighs the effort to change.

This guide is for teams approaching that point (or suspecting they might be). It's designed for engineering and business leaders who have the courage to ask,

> "Is our current CI/CD setup really serving us, or are we holding ourselves back?"

We'll walk through the signs that it might be time to re-evaluate and show you how to structure a smart, focused evaluation process that leads to an optimal decision. And when you're ready to take action, you'll get a complete RFP template and scoring framework to guide the process with clarity and speed.

Let's start with what those early signals look like, and why they're easy to ignore until they're not.

CI/CD IS MISSION-CRITICAL AND DEEPLY EMBEDDED, WHICH MAKES IT HARD TO REPLACE. Teams rarely launch a full evaluation after a single failure. Instead, performance issues like slow pipelines, test flakiness, or unclear ownership build up over time. That friction becomes hard to ignore when the organization starts to shift and your tooling can't keep up. Here are some of the most common catalysts for reevaluating CI/CD tooling.

Is it time to rethink your CI/CD platform?

Growth reveals bottlenecks

A setup that worked fine for a few teams starts to show strain as your engineering organization scales. Platform teams spend more time supporting pipelines and less time improving infrastructure. Teams begin duplicating effort. What was once fast and flexible starts to feel brittle and hard to govern.

Security and compliance raise the bar

Pressure from audits, new regulations, or security-conscious leadership can make existing CI/CD workflows feel suddenly inadequate. Visibility, access controls, artifact traceability, and policy enforcement all become harder to manage without the right tooling in place.

AI changes the development tempo

Teams using coding assistants or agent-based tooling are shipping more, faster. But that speed creates new risks. Without robust, adaptable pipelines, it's hard to keep up with code validation, testing, and security scanning. Workflows need to evolve to support automation and review at scale without slowing things down.

Platform investments need better foundations

If you're investing in internal developer platforms, golden paths, or reusable components, your CI/CD system becomes a core dependency. Legacy or VCS-bound tools often can't support the modularity, observability, or policy controls that platform teams require.

Complexity outpaces convenience

Application changes don't just happen on Git push. Infrastructure updates, container image changes, database migrations, dependency upgrades, prompt tuning, and model deployments are all part of the delivery landscape now. Teams need orchestration that works across systems, not just within them.

Modernization and cost scrutiny shift priorities

Leadership-driven mandates to modernize engineering practices, adopt platform strategies, or reduce tooling spend can expose CI/CD as a gap. What worked before may no longer align with how your teams (or your budget) are expected to operate.

Product and org change expand delivery needs

New product initiatives, increased service count, or organizational restructuring can break assumptions about how work gets done. As delivery patterns shift, CI/CD tooling that was once sufficient may no longer fit.

These changes aren't always dramatic at first, which is why many teams put off re-evaluating. But when they start stacking up, they point to the same question: **Does your CI/CD platform still fit the way your teams need to work?**

How to know when your CI/CD needs are outgrowing your tools?

IF TWO OR MORE OF THESE PATTERNS APPLY, your platform likely isn't keeping up. It may be time to begin a structured evaluation.

WHAT YOU'RE SEEING	WHAT IT LIKELY MEANS
Your current tool doesn't support the workflows your org actually needs	Your current tool doesn't support the workflows your org actually needs
Developer experience is degrading and velocity is being impacted	Developer experience is degrading and velocity is being impacted
Your CI/CD system is becoming a blocker to org-wide priorities	Your CI/CD system is becoming a blocker to org-wide priorities
Your pipelines may not be ready for the pace or complexity of modern code generation	Your pipelines may not be ready for the pace or complexity of modern code generation
You've outgrown a general-purpose tool that wasn't built to scale with your org	You've outgrown a general-purpose tool that wasn't built to scale with your org
Your platform lacks the flexibility or efficiency to meet evolving org priorities	Your platform lacks the flexibility or efficiency to meet evolving org priorities

Many teams ignore these symptoms for longer than they should, not because the problems aren't real, but because changing systems can be a daunting undertaking—especially if you don't have a high degree of certainty that your new system will deliver against your needs.

Improving CI/CD creates space for faster delivery, fewer incidents, and more time spent building. The impact is measurable. If the signals are showing up, it's time to start the conversation about your CI/CD strategy.

Building the case for change

RECOGNIZING THE NEED FOR CHANGE IS ONE THING. Getting everyone on board is another. Before launching an evaluation, make sure your key stakeholders are aligned. Not on what to adopt, but on whether the current platform is holding you back.

Four steps to alignment

1. Name the pain clearly

Document specific issues like build failures, long feedback loops, duplicate workflows, and platform team support load. Be specific and grounded in real examples.

2. Tie it to business impact

Translate the friction into outcomes leadership cares about: missed deadlines, developer attrition, incident recovery time, or slow adoption of AI workflows.

3. Find your allies

You will need voices from platform, security, and team leads. Talk to the ones already frustrated. Get their input early and bring them into the evaluation process.

4. Set the scope

Be clear about what this is. A formal evaluation of whether your current CI/CD platform can meet the needs of your organization going forward. That includes looking at alternative tools and understanding what a change would take.

You're seeking support to explore real alternatives and make an informed choice.

These steps won't get you full consensus overnight, but they give you a clear starting point. Once there's agreement that the current system deserves a closer look, you can begin shaping an evaluation process that's structured, inclusive, and built to lead to a decision.

What should your evaluation process look like?

CI/CD EVALUATIONS WORK BEST WHEN THEY'RE SYSTEMATIC.

A Request for Proposal (RFP) helps anchor the process. It captures your technical and organizational needs, helps you ask consistent questions, and gives stakeholders a shared framework for comparison. Even if you don't follow a formal procurement process, an RFP helps clarify what matters and why. A strong RFP process comes down to **four elements:**

EVALUATION TEAM

Build a small (3-5 person), cross-functional group responsible for shaping and driving the evaluation.

Evaluators may include:

- Engineering leadership
- Platform or DevEx owners
- Security, compliance, or governance leads
- Finance or procurement
- Developers familiar with current pipelines and workflows

This team anchors the process, aligns on priorities, and ensures the evaluation reflects real organizational needs.

BASELINE ASSESSMENT

Get clear on where you are before comparing options:

- Tools and scripts in active use
- Sources of developer frustration or delay
- Support and maintenance burdens for platform teams
- Gaps exposed by audits, incidents, or delivery slowdowns
- Metrics like build time, test failure rate, or onboarding duration

This baseline gives your RFP clarity and grounds vendor responses in reality.

PLATFORM REQUIREMENTS

Translate your pain points and future goals into clear, structured evaluation criteria.

- Fit with your architecture and deployment model Security, compliance, and policy enforcement needs
- Developer workflow flexibility and team autonomyEcosystem compatibility and integration requirements
- Support for platform engineering and AI-augmented development

These requirements form the backbone of your RFP and shape how you compare vendors.

GUIDING QUESTIONS

Craft targeted questions that reveal how each platform handles your most important requirements and growth plans.

- How does the platform scale with growing teams and workloads?
- What controls exist for secrets, policies, and audit visibility?
- What ecosystem integrations are supported and maintained?
- How is AI-assisted development supported or governed?

These questions help surface tradeoffs early and steer the RFP toward practical, real-world fit.

ONCE YOU HAVE THESE IN PLACE, you're ready to run a structured evaluation.
Once you have these in place, you're ready to run a structured evaluation. That starts by compiling a full RFP—a clear, shareable document that organizes your criteria and questions in a format vendors can respond to.

This does not need to be a formal procurement artifact. Think of it as a working tool that helps your team evaluate consistently and keep the decision grounded in what matters.

1. Compile the RFP document

Group your requirements and questions into themes like technical capabilities, security, developer experience, support, and commercial terms. Use a format that invites clear written responses instead of vague marketing language.

HINT: If you want a head start, the evaluation toolkit at the end of this guide includes a ready-to-use RFP template, prefilled with critical questions that surface real differences between platforms.

2. Identify serious candidates

Shortlist a few platforms that align with your technical and organizational needs.
Look past surface features and focus on how each one maps to your priorities.

3. Distribute and review

Send the RFP to selected vendors. Provide context, evaluation goals, and timelines.
Ask for written responses that speak directly to your questions.

Once responses are in, you're ready to score and compare results.

Scoring and making your decision

ONCE YOU'VE GATHERED INPUT and vendor responses, you'll need a way to compare options clearly. This is where a structured scoring model pays off.

Use a shared scoring framework

Weight your evaluation criteria based on your priorities, whether that's security, developer experience, or technical capabilities. Then score each vendor against those criteria using the answers they provided.

A shared scoring model helps normalize input across stakeholders, so a concern from the security team gets measured alongside feedback from engineering leads. This avoids recency bias, over-indexing on shiny features, or relying on the loudest voice in the room.

HINT: You'll find a complete scoring template in the evaluation toolkit linked at the end of this guide.

Get to alignment, not unanimity

You may not get full agreement. But if everyone agrees on the process and has visibility into the results, you can move forward with shared confidence, even if tradeoffs are involved.

Reduce risk with a pilot

For top contenders, run a time-boxed proof of concept. Choose real workloads, not synthetic examples. Focus on developer experience, performance at scale, and integration with existing systems.

This gives your team hands-on exposure while reducing uncertainty around migration or implementation cost.

From decision to rollout

ONCE YOU'VE CHOSEN A DIRECTION, the handoff from evaluation to implementation is just as important.

Plan for rollout in phases

Start with a few teams or services that reflect different delivery patterns. Use that to pressure test the platform, refine onboarding materials, and collect feedback.

Don't try to migrate everything at once. The goal is to build confidence and momentum, not force a switch overnight.

Invest in internal enablement

Your CI/CD platform is foundational. Give it the same support you would a new internal tool: clear documentation, golden examples, ownership, and a path for feedback.

Highlight improvements in developer experience, delivery speed, or incident response. These wins build trust and help drive adoption organically.

Measure impact over time

Look at more than pipeline speed. Are developers spending less time maintaining builds? Has platform support load gone down? Are new services getting to production faster?

These are the signs of a successful transition, and the evidence you need to show that your investment was worth it.

Your evaluation toolkit

THIS GUIDE HELPS YOU know when it's time to rethink your CI/CD tooling and how to run a structured evaluation with confidence.

To make that easier, we've prepared a complete set of resources for you:

- RFP Template: A customizable document that outlines key evaluation categories, the questions you should be asking, and a structure for comparing vendors objectively.
- Scoring Guide: A flexible model that helps you weight priorities, capture input from multiple stakeholders, and surface meaningful tradeoffs across platforms.

Everything in this guide is reflected in these tools. They are designed to save time, support better decisions, and help your team evaluate with clarity.

The best time to fix CI/CD is before it becomes the reason your team slows down. If you're ready to move forward, the tools are here to help you do it right.

circleci