GitLab

How to improve your team's productivity by

Automating Software Delivery

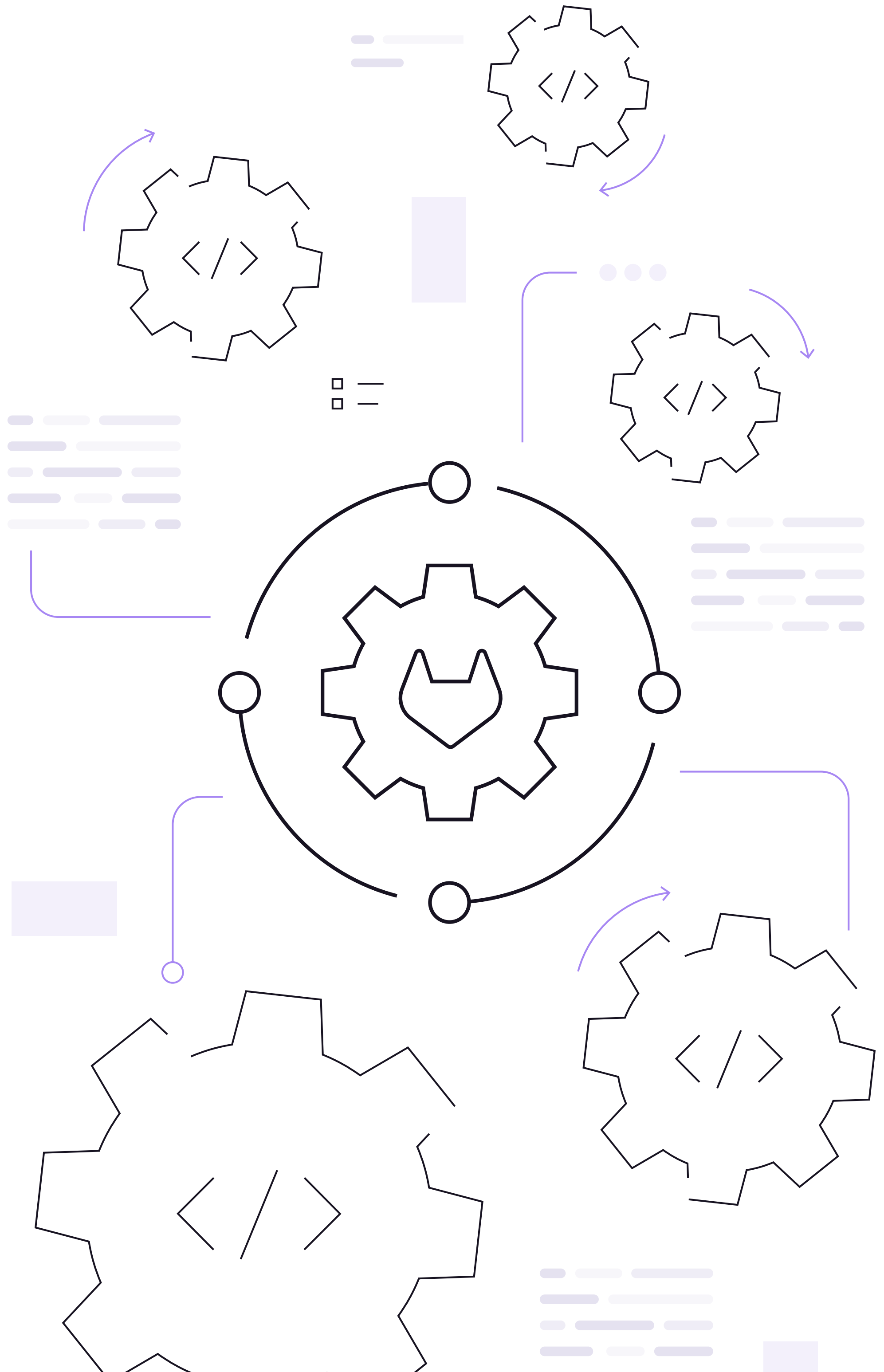


Table of contents

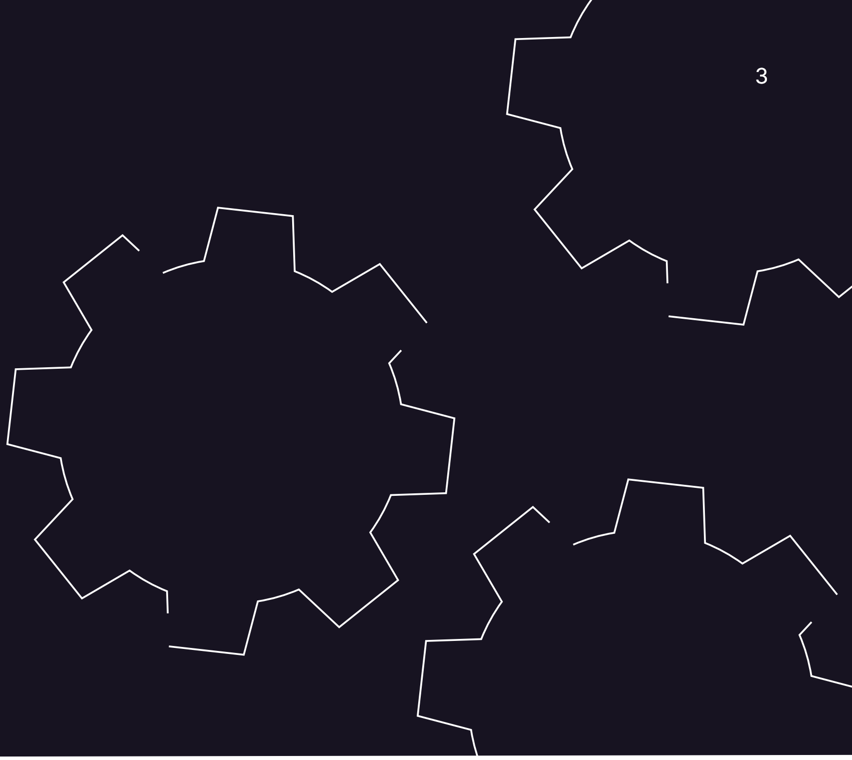
Why automating your software delivery matters

Fully automating your software delivery process can help your organization reach many of its strategic goals — and set you up to take on initiatives that may currently feel out of reach.

With automated software delivery, you can:

- Improve your team's productivity and the efficiency of your software delivery process
- Get better visibility into team performance and areas for improvement
- Enhance collaboration across teams and reduce duplicate efforts
- Give your developers a better experience
- Make every change releasable
- Improve the reliability of your applications
- Boost deployment frequency and the overall velocity of your software factory
- Speed up your cloud native and multi-cloud adoption

In this guide, we'll drill down on how you can improve productivity and efficiency with automated software delivery.



What is automated software delivery?

Automated software delivery is a set of practices that help teams deliver high-quality, secure applications more regularly and efficiently. It includes source code management, continuous integration (CI), continuous delivery (CD), automated testing, and GitOps. These are the essentials required to get started or improve the maturity of your DevOps adoption.

Automated software delivery improves productivity by...

Eliminating unnecessary manual work

At a typical company, software developers spend a sizable amount of time on repetitive, manual tasks — tasks that could be automated or eliminated with automated software delivery.

As the amount of unnecessary manual work decreases, more of the developers' time is freed up for use on more impactful and productive work. That can help improve your development velocity, the quality of your code (you're removing a lot of opportunities for human error, after all), and make your entire software delivery process more efficient.

Reducing context switching

The more manual processes and different tools there are in your software delivery lifecycle (SDLC), the more times your team will have to stop and start what they're doing, switch interfaces, wait around for things to finish, fight a fire, and more.

That's context switching. And all of it is lost productivity and efficiency — and it's probably frustrating for your team too.

With automated software delivery, you can reduce the amount of context switching your team has to do. Giving your team the freedom to focus on a single task will make them more productive and efficient, and potentially less likely to get burned out too.

Some of the manual work your team can stop doing:

- Setting up test environments
- Running test scripts
- Waiting for review comments
- Testing on local environments
- Manually creating backups
- Deploying to production

"On average, people report spending 36 minutes every day switching back and forth between applications. And they report taking on average 9 and a half minutes to get back into a good workflow once they've switched between apps."

— **Workgeist Report '21,**
a report by Qatalog and Cornell University's Ellis Idea Lab

Detecting issues earlier, when they're potentially easier to fix

With automated software delivery, you're building security and testing into the early stages of your development process. This means your team will be detecting and fixing bugs and security issues as they are created, not when code is ready to ship or already in production.

Catching issues earlier in the SDLC can make them easier and faster to fix, and makes your team more productive and efficient.

There are a few reasons for this:

1. Fixing code you wrote can be easier and more efficient when it's still fresh in your mind. When an issue isn't caught until much later, you may need to re-familiarize yourself (or someone else in your team) with the code in order to fix it.

2. If an issue isn't caught the same day it was introduced, the person who needs to resolve the issue may not be the person who wrote the code. It will inevitably take them longer to fix someone else's code, and they might have a different approach and may even try to re-write the code.

3. When you notice an issue or bug in recently pushed changes, there may be less to wade through to find and fix it. But if lots of other changes have been made since the one that introduced the problem, it can be harder to separate the issue from everything else. It will take more time to fix the issue than if it had been caught sooner, or you may decide to just roll things back to the last known good version before the bug was introduced — but then you've just undone a lot of productive work.

4. Bugs and issues that aren't caught until they're in production could have a particularly devastating effect on productivity because they may cause a full-blown fire that will likely interrupt the entire team and take multiple people to fix. In addition, you may experience an unscheduled production outage, which will affect your revenue and customers.

Plus, detecting and fixing issues earlier in the SDLC can help protect the productivity of other teams in your organization.

Protecting the productivity of your customers and other departments

The benefits of automated software delivery can extend well beyond the teams in your organization that write and ship code. As you automate your processes, you won't just be helping your team get more done — you may even be improving the productivity and efficiency of other departments in your organization, as well as the customers who use your software.

Here are a few examples:

Support

Bugs and security issues that make it into production may lead to a flood of tickets and calls for your support team. Plus, any other less urgent bug fixes — or new features — that would have made the lives of your support team easier may take longer to make their way into production.

Marketing & Public Relations

Automating your software delivery processes may help your marketing and public relations (PR) teams' productivity. Fixing issues faster and well before you're ready to deploy helps you hit your planned launch dates. This can help keep your marketing team from having to do extra unnecessary work to adjust everything they've been working on (emails, events, social media posts, press outreach, etc.) to a new timetable. And by fixing bugs and security issues before they make it out into the world, you may be keeping the marketing and PR teams from having to do damage control and communicate the issue to customers, analysts, and press. It can also be easier to market and get press to write about more exciting features and solutions that you released before your competitors.

Sales

Your sales team can be most productive and efficient when they can focus on getting potential customers to see the value of what you're selling and decide to do business with your company. If your salespeople have to spend time talking with prospects about a bug or security issue they heard about or about how the timeline for an exciting new feature has changed, that's not only unnecessary work that makes them less productive — it could also mean they'll sell less than they would have otherwise.

Customer Success

Delivering software with defects or failing to deliver software that keeps up with your competitors can encourage your customers to look for alternatives, and that could make life that much harder for your Customer Success team and anyone focused on customer satisfaction and retention.

Your customers

When you automate your software delivery, you'll likely be delivering software with fewer defects and with new features (that address customer needs and feedback) faster and more often — potentially helping customers who use your software increase their own productivity and efficiency by spending less time submitting support tickets, retrying things that should work or worked last time, and creating workarounds to get the software to do what they need. When you do what you do faster and more reliably, your customers may also be able to do so.

Improving employee retention and ability to hire

Losing valued team members is hard — especially if they're leaving for preventable reasons such as a constant pile of manual tasks or difficult-to-use legacy tools. Automating your software delivery processes means you can create a better employee experience for the teams building that software.

Here are a few ways automated software delivery can help keep your team members productive and engaged:

- Allowing the team to focus on other and perhaps more valuable work, instead of repetitive manual tasks that could be automated
- Reducing the number of fire drills from bugs and security issues that were caught late
- Giving your team more modern technology and processes to work with

And the same things that can help you *retain* members of your team can help you *recruit and hire* new ones. Savvy developers will do their research and ask questions during the interview process to suss out what the job will be like — such as what technology they'll get to use, if the company has embraced modern software development practices, and how much frustration they might run into with manual work and fire drills.

Simplifying onboarding so employees can be productive faster

The simpler it is for a new employee to learn the process and tools your team uses to develop and deploy software, the faster they can become confident and productive members of your team.

Automating your software delivery process can help you substantially cut down on the number of manual tasks and various tools your team has to learn and try to remember to do their work. This can make many other aspects of the process more efficient — especially for new employees.

Unleash your team’s productivity

Automated software delivery is your ticket to unleashing your team’s productivity and making your software development lifecycle more efficient. Free your team from manual work so they can focus on value-generating work that will get noticed.

GitLab for automated software delivery

With GitLab, you can automate your software delivery lifecycle with one single license, one permission model, one interface, and one data model to deploy software to multiple clouds — so your team can focus on creating business value instead of managing a complex toolchain.

Learn more about **automated software delivery with GitLab.**

GitLab