Scaled continuous integration & delivery

Deliver better software, faster,
with GitLab's best-in-class CI/CD

GitLab

Table of contents

CI/CD: Non-negotiable for modern software development

Today, whatever your business, customer expectations are largely the same: They want a great product and efficient service. The days of annual releases, with a pre-defined feature set bundled onto compact discs and delivered to retailers, are far behind us. Delivering customer value at a fast pace demands a refined software development lifecycle that saves time, effort, and cost wherever possible.

Continuous integration (CI) is the practice that developers use to detect, locate, and fix errors by integrating their code frequently into a shared repository and running automated tests on it. CI has become a non-negotiable aspect of everyday work for development teams.

Continuous delivery (CD) ensures that CI-verified code is always in a state that's ready to be deployed, reducing your cycle time and creating a fast, effective feedback loop between you and your customers.

Continuous deployment (CD) goes a step further simply by pushing code to production when all automated tests pass.

CI/CD plays a critical role in making sure that new code is fully functional and deployed, early and often.

We reduced our CI build queue time by 75%, which allowed developers to have test results faster and allows QA to have build artifacts to test faster.[1]

Continuous integration: is the practice of integrating code into a shared repository and building/testing each change automatically, as early as possible—usually several times a day.

[1] *GitLab 2020 Global Developer Survey, about.gitlab.com/developer-survey*

Software development before CI/CD

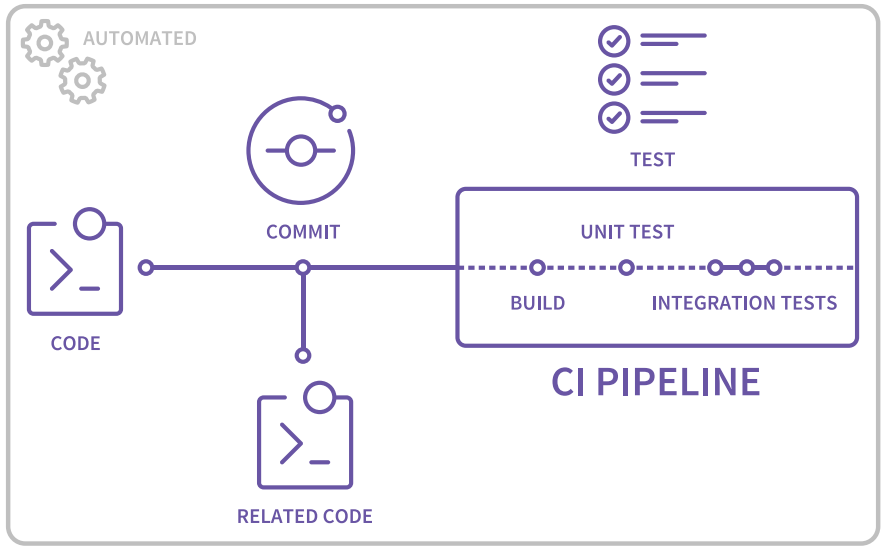
The move towards a shorter release cycle in response to the changing pace of the software landscape is enabled in many ways by adopting continuous integration, delivery, and deployment. But it wasn't always this way: Sluggish development, infrequent releases, and siloed teams created bottlenecks in the run-up to releases as merge conflicts arose late in the cycle. Responses to customer and stakeholder feedback were often stalled.

Adopting CI/CD addresses many of the technical issues responsible for inefficient software development life-cycles (such as discovering merge conflicts too late). CI/CD doesn't just have a technical impact, it influences the entire software development process, as well. From deciding on feature sets, to working on new code, to going to market and gathering user feedback to influence the next iteration, CI/CD encourages working in a more continuous way beyond the technical components of the process.

Before CI/CD	With CI/CD
Annual releases	Frequent releases
Waterfall approach	Collaborative approach
Coding in isolation	New code regularly checked into shared repository
Merge conflicts	Seamless merges
Manual testing	Automated testing
Pre-production bottlenecks	Code is production-ready at all times
Manual deployments	Automated deployments
Feedback gathered too late for next release	Rapid feedback loop

How does CI/CD work?

With a continuous integration strategy, developers check their new code frequently, triggering automated processes that occur on every commit a developer makes. Most commonly, this includes running a build and then performing automated tests, giving you a team-wide view of the health of every commit and whether it builds and passes tests. Using CI in conjunction with Git for version control, you can take advantage of fast, lightweight branching to enable building and testing changes on a staging or development branch before merging to the main branch. If all goes well, you can go ahead and merge, and perform another CI run on the main branch as an additional quality gate.



By leveraging a CI tool, development teams also take advantage of more flexibility. For example, it can run tests across multiple operating systems, multiple browsers, and, in general, do more work on a developer's machine. Waiting until the end of the development cycle to find out a change fails on a specific operating system or browser can become expensive and could jeopardize the release.

> **By leveraging CI you become proactive rather than reactive, ultimately saving money.**

A CI system can also offload a lot of work from the developer. They no longer have to worry about building their changes with a variety of different operating systems or other environments. The CI system can take care of all that, and leverage its ability to integrate with elastic compute tools like Docker and Kubernetes to take full advantage of the power of the cloud, without having to run on a development machine. Best of all, all of this work is automated, which frees up the developer to move onto other tasks while the build and tests complete.

Continuous delivery is the next step. It ensures that the staging environment is always reflective of the latest changes, so it's ready for review and feedback by stakeholders right away, and is ready to be deployed at the touch of a button. For organizations with mature DevOps practices, adopting continuous deployment is a natural progression. Automating the entire test and deploy process nets the most benefits in terms of speed of innovation.

CI/CD and your business

Sometimes the link between the tools developers want to use and the business value they offer isn't clear. With CI/CD, the connection is immediately obvious. By checking in new code regularly and testing it automatically, everyone can rest easy knowing that bug fixes, improvements and new features work and the code isn't broken, removing the anxiety from deployments.

Catching errors earlier in the cycle can prevent time-consuming and potentially costly backtracking. Ensuring that new code is always ready for instant deployment reduces bottlenecks in the run-up to a new release, so you can deliver early and often. Delivering early and often facilitates a quick feedback cycle between you and your users, boosting customer satisfaction and loyalty. And with the introduction of review apps, previewing changes, sharing them with stakeholders, and gathering feedback is as simple as sending a link to click.

A good CI/CD strategy can also generate more revenue because it reduces downtime, automates tasks, and businesses get to see the benefits of new code faster. Instead of spending time on undifferentiated engineering, developers are working on revenue-generating projects.

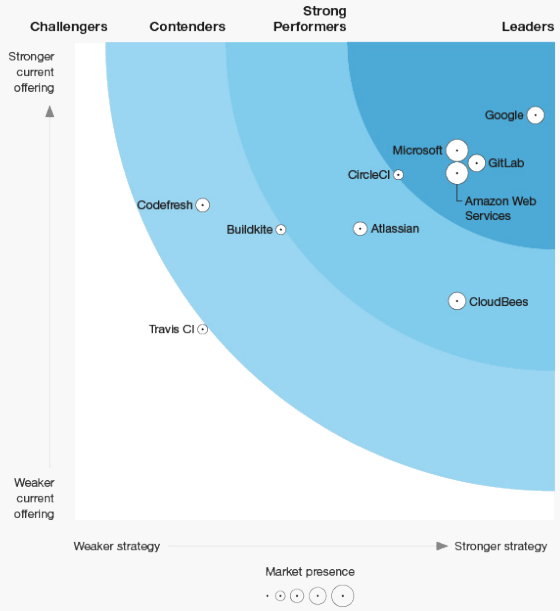
Review apps: A feature that automatically spins up a dynamic environment for merge requests, so you can preview changes right away.

Benefits of a single DevOps application

Of course, the advantages of CI/CD are neither controversial nor new, and there have been a number of solutions on the market for a while. These are usually bolted onto other tools, whether they're popular for on-premises (such as Jenkins) or rooted in the cloud (like Travis or CircleCI). Many organizations find these sufficient for their needs, but there has been growing interest in built-in CI/CD solutions. GitLab was rated as a leader in Forrester's 2019 CI Wave.

> "GitLab's simple and cohesive approach lands it squarely as a leader. GitLab's approach of having a single application to manage each phase of software development comes through in its developer experience. The UI is easy to use, cohesive, and comes integrated with collaborative tools and development tasks at a glance; it was one of the most fluid UIs we evaluated. GitLab also shines with its preconfigured environments and its ability to support on-premises build agents that can execute on multiple operating systems, including Linux, Mac, and Windows."[2]
>
> — The 2019 Forrester Wave™: Cloud-Native Continuous Integration Tools

[2] *The 2019 Forrester Wave™: Cloud-Native Continuous Integration Tools, https://about.gitlab.com/analysts/forrester-cloudci19/*



The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester's call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.

It's more time efficient

It's not unusual for an organization to rely on a number of distinct tools to fulfil each stage of the cycle, but it's not always the most efficient approach. A single application that includes agile planning, source code management, CI/CD, monitoring, and security reduces administrative complexity, and allows developers to spend less time stringing together their tooling and troubleshooting when APIs change.

Tighter integration between different stages of the development process facilitates the creation of cross-references between code, tests, and deployments while discussing them, making it easier to see the full context. Reducing the need for context-switching can also speed up workflow—leaving more time for the work that matters, like building new features.

It's more resource efficient

CI/CD coupled with elastic compute resources using containers empowers you to dynamically scale up or down easily, on demand, helping to save on infrastructure costs. If the output of your build is a container image, you can take advantage of the consistency and repeatability of Dockerizing your builds by pushing to the built-in registry and running CI from there.

It gives you more control and visibility

Having your CI/CD integrated into the repository management system gives developers more control and visibility over their build pipeline, making it even easier to identify issues early and address them while costs are still low. This is in line with the DevOps concept of "shifting left," which moves processes, such as reviews and testing to earlier stages in the software development lifecycle. The other upshot is that using version control for build scripts and CI configuration gives you the peace of mind of being able to restore easily to an earlier version if a bug is introduced.

Scaling CI/CD with containers

Containers simplify the process of testing and deploying software[3]. They allow you to package an application's code and dependencies into a single, portable "container" that can be easily moved across the development lifecycle.

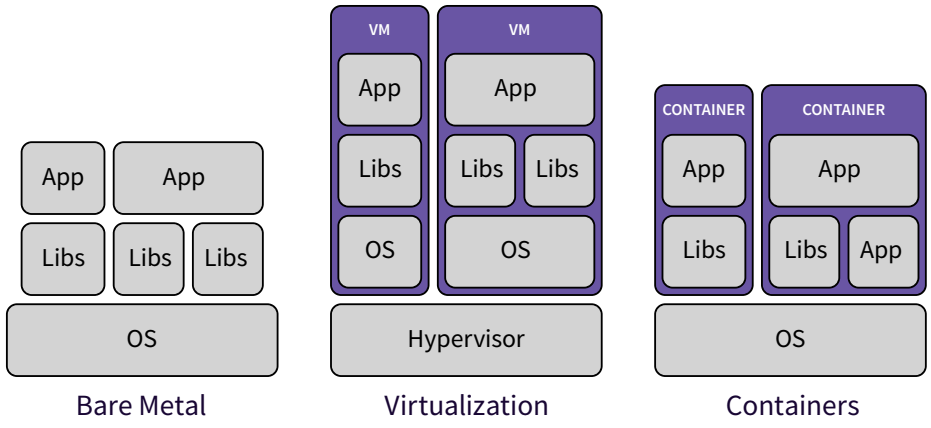
Containers facilitate CI/CD by providing reliable, secure, and efficient application environments that can easily scale up or down. Once configured, they allow you to approximate a production environment, where development teams can test the impact of their changes safely. For DevOps teams, this means that developers can find and fix errors faster, reducing dependencies on operations, and effectively speeding up the development lifecycle.

By adopting container-based application development, you can more reliably practice CI/CD.

What is a container?

A container is a method of operating system-based virtualization that allows you to securely run an application and its dependencies independently without impacting other containers or the operating system.

Containers work much like a virtual machine except that, instead of packaging your code with an operating system, containers are run as a Linux process inside of the kernel. This means that each container only holds the code and dependencies needed to run that specific application, making them smaller and faster to run.



Containers retain the same repeatability factor as virtual machines, but are much faster and use less resources to run.

[3] *Want Efficient CI and CD? Better start using containers. Chris Tozzi, TechTarget.com*

CI/CD with containers

Containers make it possible to build one version of an application that can be easily deployed to multiple environments. Code can be shipped faster when packaged in a container because errors and bugs are caught earlier in the process. Whenever new changes are introduced into the code, simply update the container image and spin up a new container. No need to uninstall the older version and replace it with a new one.

When applications are built as containers and run on an integrated CI server, developers can automatically test their code in a production like environment, closing the gap between development and operations. When integrated with other development tools, like GitLab, it's possible to automatically trigger a CI build every time a developer commits new code, run tests in parallel, and spin up and destroy ephemeral production environments to see the changes in real time.

CI/CD coupled with containers has the potential to fully automate the build, test, and deploy stages of the software development lifecycle, making it easier and faster for developers to review their code and ensure that it is production-ready.

Container: a lightweight method of operating system-based virtualization that allows you to securely run an application and its dependencies independently without impacting other applications or the operating system.

Getting started

Choosing a CI/CD solution

HOSTING

Hosting is an important consideration when choosing any new tool. Are you looking for a SaaS solution or do you want to manage your own instance? For self-managed applications you also want to be sure that installation options exist for your preferred infrastructure, whether that's an on-premises data center, a favorite cloud provider, or both. The most sophisticated CI/CD solutions can run in your data center for normal operations and then burst out to the cloud to elastically meet demand for variable load.

OPEN SOURCE VS. COMMERCIAL

Using an open source solution has its advantages: it's free of licensing costs, you can see exactly what's in the box, and make alterations if needed. However, many enterprises find the engineering resources needed to configure and maintain open source software makes the Total Cost of Ownership (TCO) significantly higher than commercial licensing costs. Make sure you do your research before committing. What if you need priority access to support? And if the vendor decides to abandon the product, can you manage without them? Ask these questions first.

SUPPORT FOR INTEGRATIONS

Find out what tools your teams are already using and if the CI/CD solution you're considering is supported. We've already mentioned the benefits of a built-in solution, and bringing all your tools under one product with one interface and datastore is also useful for things like cycle analytics, which can help to reduce the time between coming up with an idea and deploying it.

FEATURES FOR VISUALIZING THE RELEASE PROCESS

One of the advantages of leveraging CI/CD is being able to see changes and new additions from the moment they're created. Below are some of the features you'll want to look out for when choosing a CI/CD tool:

- **Review apps** - Automatically spin up dynamic environments for new code before it's merged. They are like having a staging environment for every merge request.
- Canary deployments - Minimizes the impact of any issues with a new version by deploying it to a small portion of the fleet first and monitor their usage and behavior closely. This portion serves as the "canary in the coal mine," alerting you to problems so that only a small percentage of users are affected before you rectify the error. These are just a few of the features that can make a significant difference to your team's efficiency.
- **Auto-scaling agents** - Agents, also known as "runners," are the software that runs each individual CI/CD job. The ability to elastically scale up and down is necessary to handle burst load during peak usage and save computer costs when demand lessens.

Built-in continuous integration

"For a long time GitLab CI used to be a separately deployed web application, the UIs were not integrated in any way and they felt like separate products, as if they were not even made by the same company. At one point we decided to integrate it, and literally within one or two weeks we started seeing new possibilities of interlinking these applications. We'd think, 'Hey, wouldn't it be neat if we added a button to the latest status of this page?' which previously is something we never would have thought about, because we really thought of the two as separate products that need to talk over an established channel, instead of just putting a link in everywhere where it would useful to have a link to CI. So with a built-in solution, the integration you get is not just tighter, it's integrated in ways which other, siloed development teams, developing separate products, would never think of."

— Douwe Maan, GitLab Platform Backend Lead

Adopting CI/CD

If teams are not routinely testing new code at all, adopting CI/CD can be a significant culture change. Instead of simply deploying and hoping for the best, CI/CD means a new, more proactive approach to working. This may meet some resistance up front, but the outcome of a more streamlined development cycle with less downtime and disruption is ultimately worth the effort. It can benefit teams to begin by adopting CI first and introduce CD at a later time when processes are more mature.

If you're already using a CI/CD tool and are looking to switch to another, the initial migration can take a couple of days or a couple of months depending on your migration strategy. To reduce complexity, a complete transition is recommended. For organizations with many teams and projects, start by migrating one team or project at a time.

Development teams want to work more efficiently, and organizations want to deliver better software for their users at scale. CI/CD meets both of these needs by automating the development and deployment process. This is what makes scalability possible.

For teams adopting CI for the first time, a single application like GitLab that offers source control management and continuous integration means that teams can start using CI without having to worry about a complicated toolchain that adds complexity to a brand new process.

For teams already using CI, GitLab offers industry-leading functionality that helps teams scale and innovate. Take advantage of our built-in continuous integration and continuous delivery, and have advanced support for working with containers, Kubernetes, or for developing in a microservices or even multicloud environment.

Ready to see what best-in-class CI/CD can do for your team?

Try GitLab free for 30 days

GitLab

Try GitLab free for 30 days

About GitLab

GitLab is a DevOps platform built from the ground up as a single application for all stages of the DevOps lifecycle enabling Product, Development, QA, Security, and Operations teams to work concurrently on the same project. GitLab provides a single data store, one user interface, and one permission model across the DevOps lifecycle. This allows teams to significantly reduce cycle time through more efficient collaboration and enhanced focus. Built on Open Source, GitLab leverages the community contributions of thousands of developers and millions of users to continuously deliver new DevOps innovations. More than 100,000 organizations from startups to global enterprises, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network, and Comcast trust GitLab to deliver great software faster.

GitLab is the world's largest all-remote company, with more than 1,300 team members in 68 countries and regions.

Built on open source, GitLab works alongside its growing community, which is composed of thousands of developers and millions of users, to continuously deliver new DevOps innovations. More than 100,000 organizations from startups to global enterprises, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network, and Comcast trust GitLab to deliver great software faster.


GitLab