SCALED CONTINUOUS INTEGRATION & DELIVERY

DELIVER BETTER SOFTWARE, FASTER, WITH GITLAB'S BEST-IN-CLASS CI/CD

GitLab

WHAT'S INSIDE?

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 the pace required demands a refined software development lifecycle that saves time, effort, and cost wherever possible.

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

87% of developers believe practicing continuous integration alleviates blockers in the development process[1]

Continuous Delivery (CD) is the next step: it ensures that new 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) going a step beyond simply preparing code to be deployed, continuous deployment pushes code to production when all automated tests pass.

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

Continuous Integration: The practice that developers use to detect, locate, and fix errors quickly by integrating their code frequently into a shared repository and running automated tests on it.

[1] GitLab 2018 Global Developer Survey, about.gitlab.com/developer-survey/2018

SOFTWARE DEVELOPMENT BEFORE CI/CD

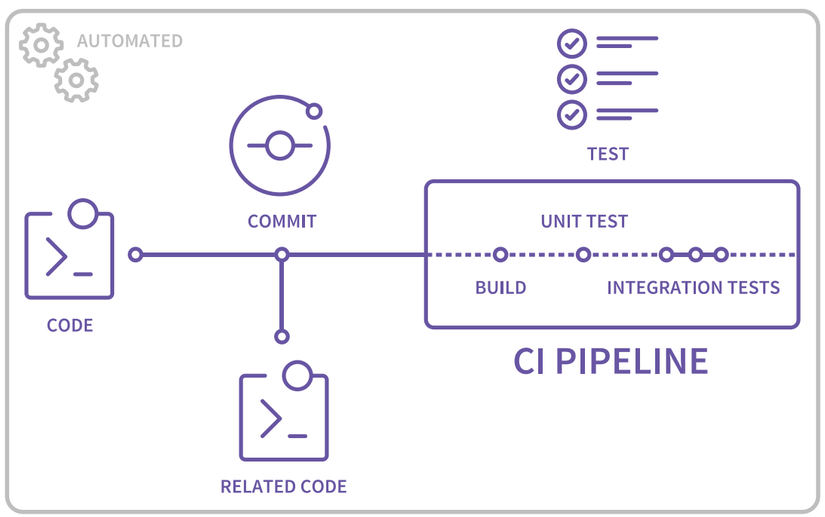
The move towards a shorter release cycle in response to the changing pace of the software landscape is enabled in many ways by adopting Continuous Integration, Delivery, and Deployment. But it wasn't always this way: sluggish development, infrequent releases and siloed teams created bottlenecks in the 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 lifecycles (such as discovering merge conflicts too late), but it doesn't just have a technical impact, it influences the entire software development process. 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 in 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, you can take advantage of fast, lightweight branching to enable building and testing changes on a staging or development branch before merging to the master branch. If all goes well, you can go ahead and merge, and perform another CI run on the master branch as an additional quality gate.



By leveraging a CI tool, you can also take advantage of the fact that it has significantly more flexibility and power available: for example, it can run tests across multiple operating systems, multiple browsers, and in general do more work than is practical on a developer's machine. If you wait until the end of the development to find out a change fails on a specific operating system or browser, that 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 an evolution of Continuous Integration: if your automated build passed all its tests, why stop there? Continuous Delivery 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. The next natural step for organizations with mature DevOps practices is to adopt Continuous Deployment. Automating the entire test and deploy process nets to most benefits in terms of speed of innovation.

CI/CD AND YOUR BUSINESS

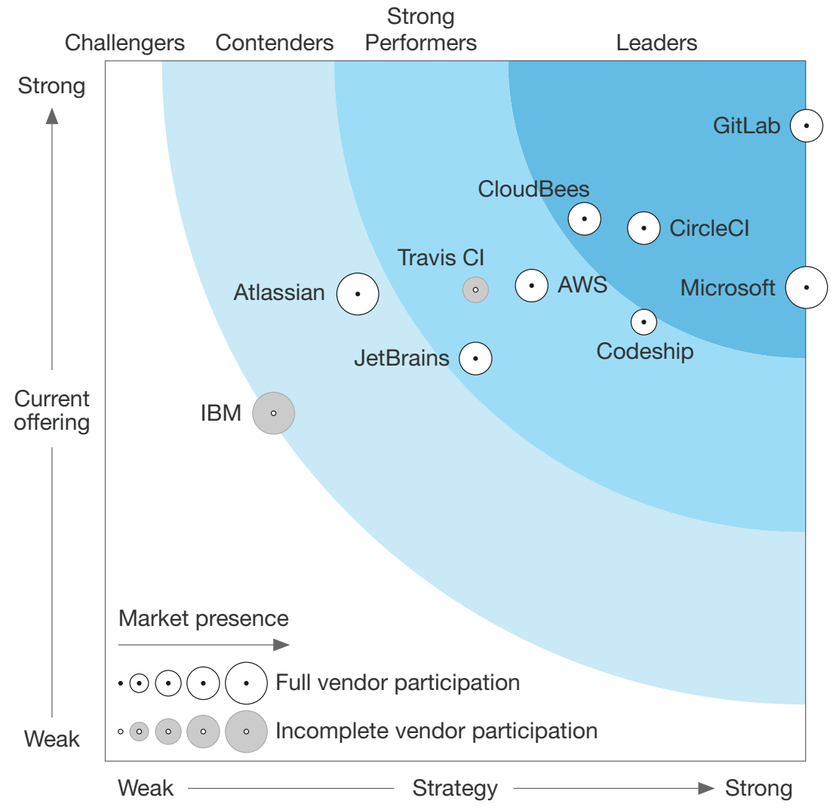
Sometimes the link between the tools developers want to use and the business value they offer isn't clear. Fortunately with CI/CD, the connection is immediately obvious. By checking in new code regularly and testing it automatically, everyone can rest easy knowing that bug fixes, improvements and new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

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 is 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 2017 CI Wave, scoring the highest marks for Current Offering as well as Strategy.



Adopting continuous integration and more agile processes can result in up to 78% savings in development costs per program, according to the HP LaserJet Firmware team.[2]

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[3]," 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.

[2] The HP FutureSmart Case Study, Dr. Nicole Forsgren, Jez Humble, and Gene Kim, ContinuousDelivery.com

[3] Shift-left quality: Getting started with DevOps metrics. TechBeacon.com

SCALING CI/CD WITH CONTAINERS

Containers simplify the process of testing and deploying software[4]. 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, allowing development teams to 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.

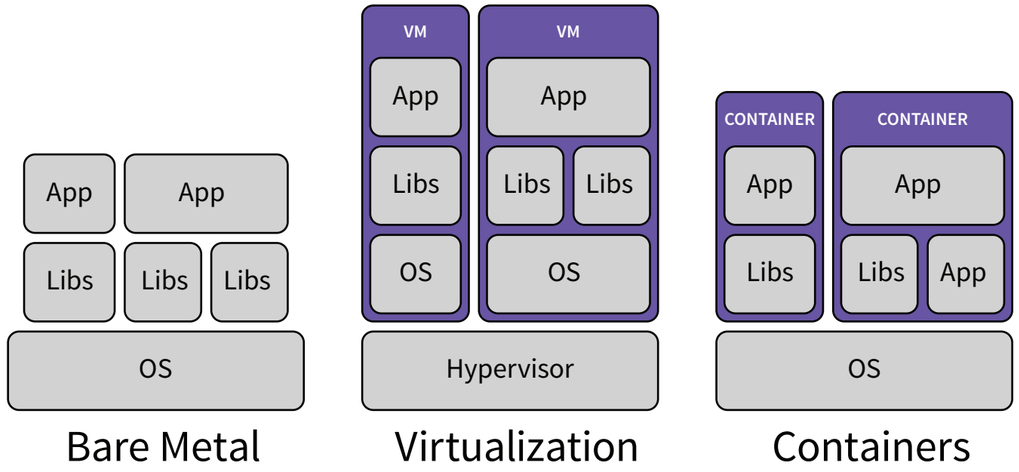
By adopting container-based application development, you can more reliably practice CI/CD.

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

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 contains 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.

CI/CD with containers

Containers make it possible to build one version of an application that can be easily deployed to multiple types of environments — whatever developers and QA run is exactly what you see in testing, staging, and production. 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, you simply update the container image and spin up a new container, versus needing to uninstall the older version and replace it with a new one.

When applications are built as containers and run on an integrated continuous integration 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 live.

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, 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.

> **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*

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

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

Canary Deployment is another popular strategy for Continuous Delivery, minimizing the impact of any issues with a new version by deploying it to a small portion of the fleet first and monitoring 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 is 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.

ADOPTING CI/CD

If you are not currently using any form of CI/CD, it may be that your teams are not routinely testing new code at all, in which case you face a significant culture change. Instead of simply deploying and hoping for the best, CI/CD means a new, more proactive approach to working, and can take some time for teams to adapt if they're not used to writing and running tests. 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. If it's less intimidating to do so, begin by adopting CI to start with and introduce CD later.

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. However, for organizations with many teams and projects, you can also start by migrating one team or project at a time.

A new tool may require slightly different processes and ways of working which can mean a shift in mindset. For example, GitLab CI/CD is more powerful if you parallelize your tests and think of them as various interdependent stages, rather than a sequential process. Teams often treat testing as a linear process so it can be a mindshift to see them rather as working simultaneously.

Have questions or need more information about getting started? We've got you covered! Get in touch.

ABOUT GITLAB

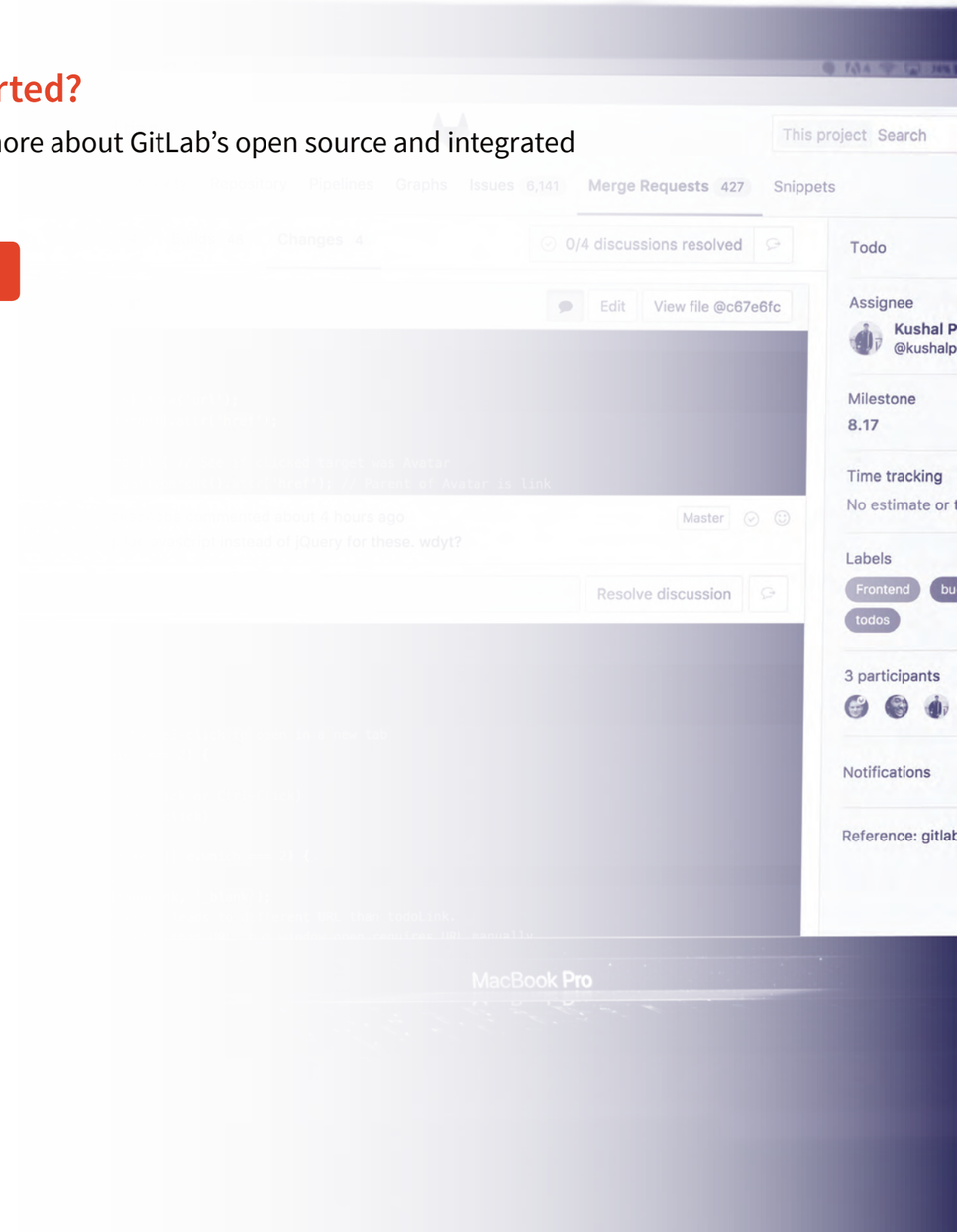
GitLab is a single application built from the ground up for all stages of the DevOps lifecycle for Product, Development, QA, Security, and Operations teams to work concurrently on the same project. GitLab provides teams a single data store, one user interface, and one permission model across the DevOps lifecycle allowing teams to collaborate and work on a project from a single conversation, significantly reducing cycle time and focus exclusively on building great software quickly. 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 enterprise organizations, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network and Comcast trust GitLab to deliver great software at new speeds.

> **“If your team is looking for a way to breathe fresh life into a legacy CI pipeline, I suggest taking a look at GitLab CI/CD. It has been a real game changer for our mobile team at Ticketmaster.”**
>
> *— JEFF KELSEY, LEAD ANDROID ENGINEER AT TICKETMASTER*
>
> *“HOW GITLAB CI/CD SUPPORTED TICKETMASTER'S RAMP UP TO WEEKLY MOBILE RELEASES”*

Ready to get started?

Get in touch to learn more about GitLab’s open source and integrated CI/CD solution.

Contact Sales




GitLab