

# What does GitLab do?

Justin  
Jan 4

1 8

## The TL;DR

GitLab is a somewhat contrarian take on [DevOps](#): it's basically one giant tool for literally anything you'd want to do relating to building and deploying software.

- DevOps spans the gamut in software, from source control to performance monitoring
- Traditionally, teams have used different tools for each part of the DevOps pipeline
- GitLab brings the entire process together with a single platform for DevOps
- Product lines include source control, issue tracking, CI/CD, and monitoring

GitLab is a very non-traditional company – beyond their unusual approach to the market, they operate completely remotely, publicly [publish their internal guidelines](#), and are open source. They also [IPOd recently](#), and are now worth \$15B – so it's an organization worth understanding.

## Refresher: what's DevOps?

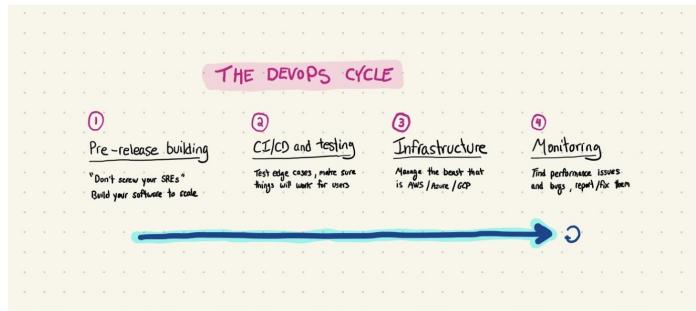
Understanding GitLab means understanding DevOps, which is thankfully the subject of [this recent Technically post](#). Here's a quick refresher if you can't be bothered to click.

Much of the tedious process of building software is actually what happens *after* you write working code: testing that code to make sure it's going to work for your users, and then actually getting it out there into the wild, and monitoring it to make sure it's performing well. These three pieces are often categorized as:

- [CI](#), or Continuous Integration – testing your code frequently
- [CD](#), or Continuous Deployment – deploying your code frequently
- [APM](#), or Application Performance Monitoring – keeping an eye on your app's performance

Imagine you're an engineer building a new feature for your company. You get the code working on your laptop and acting as it's supposed to. Now what? You've got two big steps left before your users can get a hold of it. First, you need to merge that code into the rest of the codebase, including testing it heavily across a few dimensions to make sure it doesn't break anything. Second, you need to push that updated codebase to your application's users. Finally, once it's in the wild, you want to make sure it performs well and doesn't break, for eternity.

This is basically what DevOps is, the operations of shipping software. And it's really a cycle, since this process happens all the time:



A common "gotcha" when you hear the word "DevOps" – it could be referring to any number of permutations on the above idea: a philosophy, a series of tools, a process, you name it. The definition we're using here relates to workflows and tooling, which is the one you need to understand GitLab.

## GitLab's philosophy: one stop shop

With that in mind, it's pretty easy to understand what GitLab does – they provide tools for **every step** of the DevOps process, from issue tracking to monitoring.

The norm – for sure at small to medium size companies – is that each part of this DevOps workflow requires a disparate tool. A normal stack might look something like this:

- *Issue tracking*
  - **What it is:** project management software for software engineering.  
Keep track of bugs and new features to build.
  - **What companies use:** [JIRA](#), [Linear](#), [Shortcut](#)
- *Source control*
  - **What it is:** [hosting for your code](#), with features around branching and merging.
  - **What companies use:** [GitHub](#), [Bitbucket](#)
- *CI/CD*
  - **What it is:** [testing and deploying your code](#) frequently and securely.
  - **What companies use:** (too many to name) [CircleCI](#), [Jenkins](#), [TravisCI](#), [Harness](#), [JFrog](#), etc.
- *Monitoring*
  - **What it is:** monitoring your app's performance and scalability.
  - **What companies use:** [Datadog](#), [Grafana](#), etc.

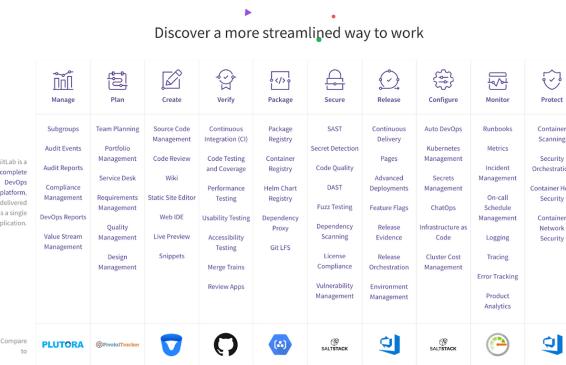
Phew, that's a lot – and this is just for DevOps. Though there are benefits to customizing your stack with purpose-built tools for specific tasks, it's not hard to see that this can become annoying over time: each piece of software requires its own integrations, maintenance, and billing scheme.

#### 🔍 Deeper Look 🔎

Buying, integrating, and maintaining software – especially in developer tools, and especially at larger companies – can be a surprisingly resource intensive endeavor. Integrating a tool like Datadog requires changes to your application code, e.g. [using tracers in Node](#). While the setup costs are usually up front, they can be non-negligible, hence GitLab's pitch.

#### 🔍 Deeper Look 🔎

GitLab's take is that you should be doing all of this in one, single tool to consolidate your stack. This comical visualization on their [homepage](#) puts things into perspective:



With that in mind, let's dive a bit deeper into what their product offerings look like.

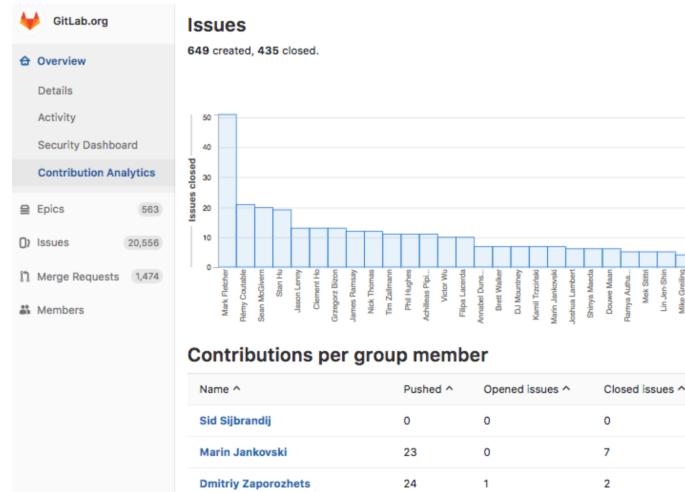
## GitLab's product lines

GitLab wants you to use them for everything you'd do that relates to DevOps, from tracking what you're working on all the way to monitoring your application's performance. We'll cover the important ones:

### 1. Issue tracking

You can use GitLab to [project manage software engineering](#), like you'd use [Jira](#). Each feature you want to build, bug you need to fix, or even small cosmetic "let's

get to this when we have time" things get what's called an *issue* or a *ticket*. If you've heard the phrase "pick a ticket off the backlog" from an engineer, that's what they're referring to.



Like other tools, GitLab also gives you analytics on top of your issues: how fast things are getting built, who is contributing the most, and things of that nature.

*Alternatives:* Jira, Linear, Shortcut

## 2. Source / version control

You can use GitLab to manage your code repositories and handle merging in changes from your team, like you'd use GitHub. The actual application code is hosted on a GitLab server, and developers pull local copies to their laptops to develop new features or fix bugs. Once they're ready to merge that new code in, you use what GitLab calls their Merge Request feature, loosely comparable to a GitHub pull request.

**Repository**

Files  
Commits  
Branches  
Tags  
Contributors  
Graph  
Compare  
Charts  
Locked Files

**Issues** 10,499

**Merge branch '45481-sane-pages-artifacts' into 'main'**  
Kamil Trzcinski authored about 2 hours ago

Name	Last commit
.github	Address
.gitlab	Fix link i
app	Merge b
bin	add initi
builds	Add mis
changelogs	Merge b
config	Merge b

*Alternatives:* GitHub, BitBucket

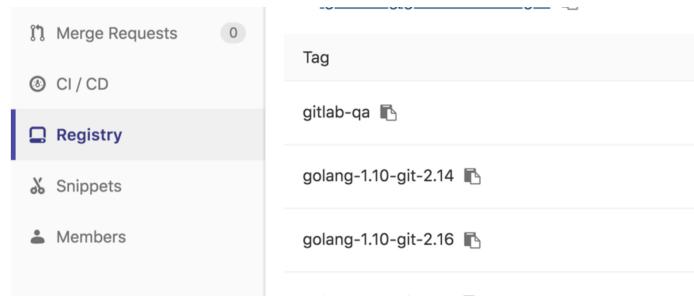
## 3. Package / container registry

GitLab provides a utility for packaging and storing any **dependencies** that your application relies on. For example, if you've built your web application in React, you need somewhere to store the actual code that Facebook wrote to build React. Most teams use a package manager like npm, but for more mission critical applications with larger packages, teams will use something like GitLab to store and host those packages.

**Container Registry**

With the Docker Container Registry integrated into GitLab, you can now store and share your Docker images directly from within the GitLab interface. Learn more about [Container Registry](#).

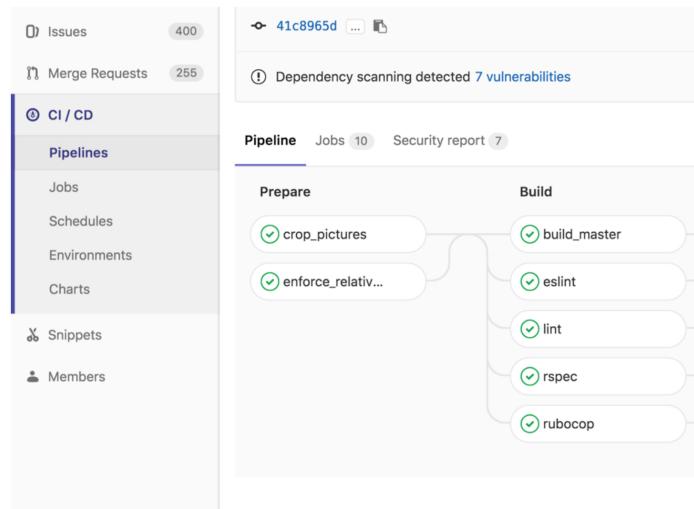
[gitlab-org/gitlab-build-images](#)



*Alternatives:* JFrog, npm

#### 4. CI/CD

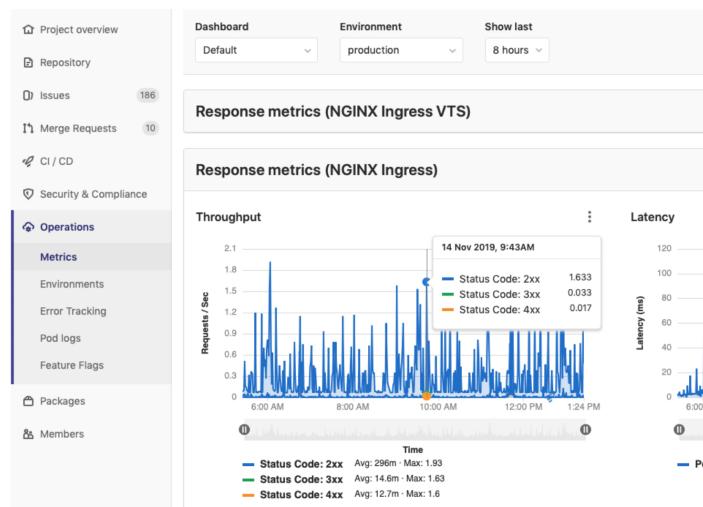
You can use GitLab to run your [CI/CD pipelines](#). For CI, you might use GitLab to run unit tests that make sure your code operates as intended when you create a merge request. For CD, you might use GitLab to run your app's build steps and release it to your users.



*Alternatives:* CircleCI, Harness, JFrog

#### 5. Monitoring

Once you've gotten your application out to your users, you can use GitLab to monitor its performance, like you might use [Datadog](#). Teams commonly look at metrics related to speed, i.e. how quickly APIs respond to requests, as well as reliability, i.e. how often requests return successful responses and if any errors are unusually frequent.



*Alternatives:* Datadog, Grafana, Elastic

Finally, some closing notes on GitLab. This is an enterprise company who makes

their money through large deployments at slow moving, massive companies. And this is why many people have a hard time understanding what they do.

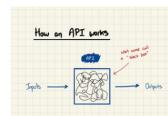
If you've been reading Technically for the past year, you're familiar with the Technically Sniff Test™ – if the company's website **does not** explain what their product does, they're focusing on enterprises. And GitLab is certainly an offender here. The majority of the homepage speaks of faster iteration, business value, transformation, and innovation cycles (?). Documentation is nested two levels deep in their navigation. There are webcasts (?) and e-books advertised above the fold. This is a company selling to executives, not developers.

8    Comment    Share

Write a comment...

[Top](#)   [New](#)   [Community](#)

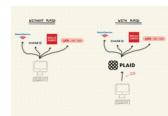
Q



### What's an API?

What McDonalds and Lyft have in common

Justin Jan 9, 2020 187 0



### What does Plaid do?

Technically begrudgingly tackles Fintech

Justin Jan 14, 2021 34 3



### What does New Relic do?

Keeping an eye on your servers and apps

Justin Jan 11 23 2



### What's Reverse ETL?

Getting your data OUT of your warehouse?

Justin Feb 1 19 0



### What happened to Facebook?

A basic explainer of what that outage was all about

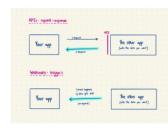
Justin Oct 5, 2021 29 4



### What's DevOps?

IT has a cool new name

Justin Jan 5, 2021 34 0



### What are webhooks?

Triggered

Justin Sep 13, 2021 28 5



### What's Headless E-Commerce?

We may be running out of names

Justin Nov 2, 2021 20 7



### I built a (basic) Substack clone in a month

This was probably a waste of my time

Justin Sep 2, 2020 50 4



### What's Kafka and what does Confluent do?

Help with solving Kafka-esque data problems

Justin Jun 15, 2021 29 4

[See all >](#)

© 2022 Justin · [Privacy](#) · [Terms](#) · [Collection notice](#)

 [Publish on Substack](#)

[Our use of cookies](#)

Substack is the home for great writing  
We use necessary cookies to make our site work. We also set performance and functionality cookies that help us make improvements by measuring traffic on our site. For more detailed information about the cookies we use, please see our [privacy policy](#).