Software Engineering Intern
at GitLab
Remote
Software Engineering Interns will work on the Gitlab Product, like all engineers who work on the product. This may include both the open source version of GitLab, the enterprise editions, and the GitLab.com service. You will work in either a Frontend or Backend team on a mature stage product at Gitlab.

Requirements

- Experience coding with Ruby on Rails or JavaScript framework (ideally Vue.js or other JS frameworks) either through university, personal projects or previous work experience.
- Proficiency in the English language, both written and verbal, sufficient for success in a remote and largely asynchronous work environment
- The ability to understand complex technical, architectural, and/or organizational problems
- Comfort working in a highly agile, intensely iterative software development process
- Positive and solution-oriented mindset
- Effective communication skills: Regularly achieve consensus with peers, and clear status updates
- An inclination towards communication, inclusion, and visibility
- Self-motivated and self-managing, with strong organizational skills, but with plenty of support from mentors, managers and internship managers throughout the programme
- Share our values, and work in accordance with those value
- Ability to thrive in a fully remote organization
- Computer Science Education or equivalent courses
- Contributor to open source projects

Nice-to-haves

- Experience with the GitLab product as a user or contributor
- Completed foundation studies (1st year of college/university) in Computer Science or equivalent courses

Responsibilities

- Develop features and improvements to the GitLab product in a secure, well-tested, and performant way
- Collaborate with Mentors and your Manager to maintain a high bar for quality in a fast-paced, iterative environment
- Advocate for improvements to product quality, security, and performance
- Solve technical problems of varying level of complexity
- Craft code that meets our internal standards for style, maintainability, and best practices for a high-scale web environment.
- Confidently ship small features and improvements with guidance and support from mentors and managers.
-

Hiring Process

Due to the high volume of applications, the bar is set very high and applications will be looked at extensively when making a decision to move a candidate forward.

1. Application

    1. Candidates are required to provide answers to:
        1. Technical Questions
        2. Situational Questions rather than STAR, as may not have any direct work/relatable experience.
        3. Cover Letter addressing 1-2 specific topics
2. Technical Assessment
    1. Automated online technical assessment
    2. Technical assessment interview
3. Behaviourial and Values based panel interview consisting of 3 panelists:
    1. Hiring Manager
    2. Recruiter/HR
    3. Misc Interviewer
4. Decision is made and successful candidates will subsequently be made an offer. Additional details about our process can be found on our hiring page

Questions
Write about a project that you worked on that you are proud of. How did this project become a success and what was your role in the success? *

Write about a time when you started something and had to take a course correction. Why did you do this? What were the consequences and successes? *

Write about a time that you failed at something, what did you learn? *

Being a pretty standard Rails application, GitLab is built using the MVC design pattern. Please describe in as much detail as you think is appropriate what the responsibilities of the Model, View, and Controller are, and what the benefits of this separation are. *

A user browses to https://gitlab.com/gitlab-org/gitlab in their browser. Please describe in as much detail as you think is appropriate the lifecycle of this request and what happens in the browser, over the network, on GitLab servers, and in the GitLab Rails application before the request completes. *

Tell us about your latest "hard to debug" code problem you encountered. How did you resolve it? Which tools did you use *

https://about.gitlab.com/handbook/hiring/

Definitions

Job families and vacancies are different things and can't be used interchangeably. Hopefully this information will help to clarify the differences.

- A job family is a permanent item; its content is a superset of all vacancies for the job family, and it is created with a merge request. Since it is permanent please don't include text that becomes outdated when we hire someone, for example: "We are seeking".
- A vacancy is a temporary item posted on Greenhouse; its content is a subset of the job family, and it is created by copying parts of a job family based on an issue.

We don't use the word "job" to refer to a job family or vacancy because it is ambiguous.

People at GitLab can be a specialist on one thing and expert in many:

- A specialization is specific to a job family, each team member can have only one, it defined on the relevant job family page. A specialist uses the compensation benchmark of the job family.
- An expertise is not specific to a job family, each team member can have multiple ones, the expertises a team member has are listed on our team page.

The example below shows how we describe what someone does at GitLab:

```
1. Level: Senior
1. Job family: Developer
1. Specialist: Gitaly specialist
1. Location: EMEA
1. Expert: Reliability, Durability
```

We use the following terms to refer to a combination of the above:

Title

Level and job family as listed on the contract, for example: Senior Developer

Headline

All parts except expertise, as listed on vacancies, for example: Senior Developer, Gitaly specialist, EMEA

Please use the same order as in the examples above, a few notes:

- Level comes before job family.
- Specialist comes after job family and always includes 'specialist'.
- Location comes after specialization.

We preface a title with "interim" when we're hiring for the position.

Equal Employment Opportunity

Diversity & Inclusion is one of GitLab's core values and GitLab is dedicated to providing equal employment opportunities (EEO) to all team members and candidates for employment without regard to race, color, religion, gender, national origin, age, disability, or genetics. One example of how put this into practice is through sponsorship of diversity events

GitLab complies with all applicable laws governing nondiscrimination in employment. This policy applies to all terms and conditions of employment, including recruiting, hiring, placement, promotion, termination, layoff, recall, transfer, leaves of absence, compensation, and training. GitLab expressly prohibits any form of workplace harassment. Improper interference with the ability of GitLab's team members to perform their role duties may result in discipline up to and including discharge. If you have any complaints, concerns, or suggestions to do better please contact People Business Partner.

https://about.gitlab.com/company/strategy/

Organization

GitLab's mission, vision, strategy, and planning follow a cadence. Along each time period, we answer various fundamental questions: why, what, how, and when. The matrix of questions vs time period is below, with mappings to the appropriate sections of this page.

Time Period	30 Years	10 Years	3 Years	1 Year
Why	Why			
What	Mission	Vision, BHAG, Goals	Strategy	
How	Values	How	Principles, Assumptions, Pricing, Challenges, Dual Flywheels, KPIs	
When			Sequence	Plan

Why

We believe in a world where **everyone can contribute**. We believe that all digital products should be open to contributions; from legal documents to movie scripts, and from websites to chip designs.

Allowing everyone to make a proposal is the core of what a DVCS (Distributed Version Control System) such as Git enables. No invite needed: if you can see it, you can contribute.

We think that it is logical that our collaboration tools are a collaborative work themselves. More than 2,000 people have contributed to GitLab to make that a reality.

Mission

Therefore, it is GitLab's mission to change all creative work from read-only to read-write so that **everyone can contribute**.

When **everyone can contribute**, consumers become contributors and we greatly increase the rate of human progress.

Values

Our mission guides our path, and we live our values along this path.

Vision

In summary, our vision is as follows:

GitLab Inc. develops great open source software to enable people to collaborate in this way. GitLab is a single application based on convention over configuration that everyone should be able to afford and adapt. With GitLab, **everyone can contribute**.

Big Hairy Audacious Goal

Our BHAG is to become the most popular collaboration tool for knowledge workers in any industry. For this, we need to make the DevOps lifecycle much more user friendly.

How

Everyone can contribute to digital products with GitLab, to GitLab itself, and to our organization.

1. To ensure that **everyone can contribute with GitLab** we allow anyone to create a proposal, at any time, without setup, and with confidence. Let's analyze that sentence a bit.

    - Anyone: Every person in the world should be able to afford great DevOps software. GitLab.com has free private repos and CI runners and GitLab CE is free as in speech and as in beer. But open source is more than a license, that is why we are a good steward of GitLab CEand keep both GitLab CE and EE open to inspection, modifications, enhancements, and suggestions.
    - Create: It is a single application based on convention over configuration.
    - Proposal: With Git, if you can read it, you can fork it to create a proposal.
    - At any time: you can work concurrently to other people, without having to wait for permission or approval from others.
    - Without setup: You can make something without installing or configuring for hours with our web IDE and Auto DevOps.
    - With confidence: Reduce the risk of a flawed proposal with review apps, CI/CD, code quality, security scans, performance testing, and monitoring.

2. To ensure that **everyone can contribute to GitLab, the application** we actively welcome contributors. We do this by having quality code, tests, documentation, popular frameworks, and offering a comprehensive GitLab Development Kit and a dedicated GitLab Design System. We use GitLab at GitLab Inc., we dogfood it and make it a tool we continue to love. We celebrate contributions by recognizing a Most Valuable Person (MVP) every month. We allow everyone to anticipate, propose, discuss, and contribute features by having everything on a public issue tracker. We ship a new version every month so contributions and feedback are visible fast. To contribute to open source software, people must be empowered to learn programming. That is why we sponsor initiatives such as Rails Girls.
   There are a few significant, but often overlooked, nuances of the **everyone can contribute to GitLab, the application** mantra:

    - While collaboration is a core value of GitLab, over collaborating tends to involve team members unnecessarily, leading to consensus-based decision making, and ultimately slowing the pace of improvement in the GitLab application. Consider doing it yourself, creating a merge request, and facilitating a discussion on the solution.
    - For valuable features in line with our product philosophy, that do not yet exist within the application, don't worry about UX having a world class design before shipping. While we must be good stewards of maintaining a quality product, we also believe in rapid iteration to add polish and depth after an MVC is created.
    - Prefer creating merge requests ahead of issues in order to suggest a tangible change to facilitate collaboration, driving conversation to the recommended implementation.
    - Contributors should feel free to create what they need in GitLab. If quality engineering requires charting features, for example, which would normally be implemented out of another team, they should feel empowered to prioritize their own time to focus on this aspect of the application.

    - GitLab maintainers, developers, and Product Managers should be viewed as coaches for contributions, independent of source. While there are contributions that may not get merged as-is (such as copy/paste of EE code into the CE code base or features that aren't aligned with product philosophy), the goal is to coach contributors to contribute in ways that are cohesive to the rest of the application.

3. A group discussion reiterating the importance of everyone being able to contribute:

4. To ensure that **everyone can contribute to GitLab the company** we have open business processes that allow all team members to suggest improvements to our handbook. We hire remotely so everyone with an internet connection can come work for us and be judged on results, not presence in an office. We offer equal opportunity for every nationality. We are agnostic to location and create more equality of opportunity in the world. We engage on Hacker News, Twitter, and our blog post comments. And we strive to take decisions guided by our values.

Goals

1. Ensure that **everyone can contribute** in the 3 ways outlined above.

2. Become most used software for the software development lifecycle and collaboration on all digital content by following the sequence below.

3. Complete our product vision of a single application based on convention over configuration.

4. Offer a sense of progress in a supportive environment with smart colleagues.

5. Stay independent so we can preserve our values. Since we took external investment, we need a liquidity event. To stay independent, we want to become a public company instead of being acquired.

Strategy

Along the road to realizing our mission of **everyone can contribute**, our strategic goal is to become the leading, if not the only, complete DevOps platform delivered as a single application. We believe we can achieve this due to the dual flywheels of our open-core model.

More detail on our product strategy can be found on our direction page.

Sequence

Our strategy is on a 3 year cadence.

Our near team goal is to go public **in CY2020**, specifically on Wednesday November 18, 2020 which is five years after the first people got stock options with 4 years of vesting. November 18th, 2020 was also set for a number of other fun, company culture related reasons.

By November 18, 2023 we want to:

1. **Grow Incremental ACV**, resulting in over $1B ARR, and be cash flow positive.
2. **Popular next generation product** where 50% of our categories are at lovable maturity and with 100m SMAU.

3. **Highly performant team** with top talent team-member satisfaction above 90% and more than 90% of top talent answering "My team is a highly performing team." in the affirmative.

The top 3 objectives of our OKRs have matched the bold part of this strategy since we started our OKRs in CY2017-Q3.

Breadth over depth

We realize our competitors have started earlier and have more capital. Because we started later we need a more compelling product that covers the complete scope with a single application based on convention over configuration in a cloud native way. Because we have less capital, we need to build that as a community. Therefore it is important to share and ship our vision for the product. The people that have the most knowledge have to prioritize **breadth over depth** since only they can add new functionality. Making the functionality more comprehensive requires less coordination than making the initial minimal feature. Shipping functionality that is incomplete to expand the scope sometimes goes against our instincts. However leading the way is needed to allow others to see our path and contribute. With others contributing, we'll iterate faster to improve and polish functionality over time. So when in doubt, the rule of thumb is breadth over depth, so everyone can contribute.

If you want an analogy think of our product team as a plow way in front that tills the earth. It takes a while for the plants (complete features) to grow behind it. This tilled earth is ugly to look at but it surfaces the nutrients that the wider community needs to be inspired and to contribute.

If we can make a product that is strong with all features from planning to monitoring, and it works well, then we believe we can become the number one solution that companies standardize around. We need to offer the benefits that you can only have with an integrated product.

So breadth over depth is the strategy for GitLab the company. GitLab the project should have depth in every category it offers. It will take a few years to become best in class in a certain space because we depend on users contributing back, and we publish that journey on our maturity page. But that is the end goal, an application of unmatched breadth and depth.

Principles

1. Independence: since we took financing we need to have a liquidity event; to maintain independence we want to become a public company rather than be acquired.
2. Low burn: spend seed money like it is the last we'll raise, maintain 2 years of runway.
3. First time right: last to market so we get it right the first time, a fast follower with taste.
4. Values: make decisions based on our values, even if it is inconvenient.
5. Reach: go for a broad reach, no focus on business verticals or certain programming languages.
6. Breadth over depth: See Breadth over depth.
7. Speed: ship every change in the next release to maximize responsiveness and learning.
8. Life balance: we want people to stay with us for a long time, so it is important to take time off, work on life balance, and being remote-only is a large part of the solution.

Assumptions

1. Open source user benefits: significant advantages over proprietary software because of its faster innovation, higher quality, freedom from vendor lock-in, greater security, and lower total cost of ownership.

2. Open Source stewardship: community comes first, we play well with others and share the pie with other organizations commercializing GitLab.

3. Innersourcing is needed and will force companies to choose one solution top-down.

4. Git will dominate the version control market in CY2020.

5. A single application where interdependence creates exceptional value is superior to a collection of tools or a network of tools. Even so, good integrations are important for network effects and making it possible to integrate GitLab into an organization.

6. To be sustainable we need an open core model that includes a proprietary GitLab EE.

7. EE needs a low base price that is publicly available to compete for reach with CE, established competitors, and new entrants to the market.

8. The low base price for EE is supplemented by a large set of options aimed at larger organizations that get a lot of value from GitLab.

Pricing

Most of GitLab functionality is and will be available for free in Core. Our paid tiers include features that are more relevant for managers, directors, and executives. We promise all major features in our scope are available in Core too. Instead of charging for specific parts of our scope (CI, Monitoring, etc.) we charge for smaller features that you are more likely to need if you use GitLab with a lot of users. There are a couple of reasons for this:

1. We want to be a good steward of our open source product.
2. Giving a great free product is part of our go to market, it helps create new users and customers.
3. Having our scope available to all users increases adoption of our scope and helps people see the benefit of an single application.
4. Including all major features in Core helps reduce merge conflicts between CE and EE

Because we have a great free product we can't have one price. Setting it high would make the difference from the free version too high. Setting it low would make it hard to run a sustainable business. There is no middle ground that would work out with one price.

That is why we have a Starter, Premium, and Ultimate tiers. The price difference between each of them is half an order of magnitude (5x).

We charge for making people more effective and will charge per user, per application, or per instance. We do include free minutes with our subscriptions and trials to make it easier for users to get started. As we look towards more deployment-related functionality on .com it's tempting to offer compute and charge a percent on top of, for example, Google Cloud Platform (GCP). We don't want to charge an ambiguous margin on top of another provider since this limits user choice and is not transparent. So we will always let you BYOK (bring

your own Kubernetes) and never lock you into our infrastructure to charge you an opaque premium on those costs.

Customer acceptance

We firmly adhere to laws including trade compliance laws in countries where we do business, and welcome everyone abiding by those legal restrictions to be customers of GitLab. In some circumstances, we may opt to not work with particular organizations, on a case-by-case basis. Some reasons we may choose not to work with certain entities include, but are not limited to:

1. Engaging in illegal, unlawful behavior.
2. Making derogatory statements or threats toward our community.
3. Encouraging violence or discrimination against legally protected groups.

This policy is in alignment with our mission, contributor and employee code-of-conduct and company values. Here are some links that may give you some background at how we arrived at this customer acceptance policy:

- Our mission is "everyone can contribute." This mission is in alignment with our open source roots and the MIT license our open source software is subject to. The MIT license is a free software license that allows the freedom to run the program as you wish, for any purpose. This informs our customer acceptance policy.

- GitLab has a contributor code of conduct for *how* to contribute to GitLab, but there are no restrictions on *who* can contribute to GitLab. We desire that everyone can contribute, as long as they abide by the code of conduct.

- GitLab has a set of values for how Gitlabbers strive to conduct themselves. We don't expect all companies to value collaboration, results, efficiency, diversity, inclusion and transparency in the same way we do. As an open company, "everyone can contribute" is our default and transparencyis our check and balance. Transparency means our handbook, issues, merge requests and product roadmap are online for everyone to see and contribute to.

- Related topic: At GitLab, we want to avoid an environment where people feel alienated for their religious or political opinions. Therefore, we encourage Gitlabbers to refrain from taking positions on specific religious or political issues in public company forums (such as on the GitLab Contribute stage, the daily company call or Slack channels) because it is easy to alienate people that may have a minority opinion. It is acceptable to bring up these topics in social contexts such as coffee chats and real-life meetups with other coworkers, but always be aware of cultural sensitivities, exercise your best judgement, and make sure you stay within the boundaries of our code of conduct. We always encourage discussion and iteration on any company policy, including this one.

Challenges as we grow and scale

Losing the interest of the open source community would be detrimental to our success. For example, if someone wanted to make a contribution to CE and decided not to merge it because a similar feature already existed in EE, we would have lost out on an important contribution from the community.

We'll also need to adapt with a changing market. Netflix is a great example of this. Everyone knew that video on demand was the future. Netflix, however, started shipping DVDs over mail. They knew that it would get them a database of content that people would want to watch on demand. Timing is everything.

If a new, better version control technology dominates the market, we will need to adopt it and keep an open mind. Hopefully, we will be big enough at that point that people will consider us an integrated DevOps product. If not, we can always change our name, but we are currently investing to make git better for everyone.

For more thoughts on pricing please see our pricing model.

Dual flywheels

GitLab has two flywheel strategies that reinforce each other: our open core flywheel and our development spend flywheel. A flywheel strategy is defined as one that has a positive feedback loops that build momentum, increasing the payoff of incremental effort. You can visualize how the flywheels work in congruence via the diagram below. The KPI and responsibilities table lists the relevant indicator and department for every part of the flywheel.

Part of flywheel	Key Performance Indicator (KPI)	Department
More Users	Stage Monthly Active Users	Marketing
More Contributions	Wider community contributions per release	Community Relations
More Features	Merge Requests per release per engineer in product development	Engineering and Product Management
More Revenue	IACV vs. plan	Sales

Publicly viewable OKRs and KPIs

To make sure our goals are clearly defined and aligned throughout the organization, we make use of Objectives and Key Results (OKRs) and Key Performance Indicators (KPIs) which are both publicly viewable.

Plan

Our near terms plans are captured on our direction page.

Why is this page public?

Our strategy is completely public because transparency is one of our values. We're not afraid of sharing our strategy because, as Peter Drucker said, "Strategy is a commodity, execution is an art."

https://about.gitlab.com/handbook/values/
CREDIT

GitLab's six values are **Collaboration**, **Results**, **Efficiency**, **Diversity & Inclusion**, **Iteration**, and **Transparency**, and together they spell the **CREDIT** we give each other by assuming good intent. They are made actionable below.

About our Values

We take inspiration from other companies, and we always go for the boring solutions. Just like the rest of our work, we continually adjust our values and strive always to make them better. We used to have more values, but it was difficult to remember them all, so we condensed them and gave sub-values and created an acronym. Everyone is welcome to suggest improvements.

Collaboration

Helping others is a priority, even when it is not immediately related to the goals that you are trying to achieve. Similarly, you can rely on others for help and advice—in fact, you're expected to do so. Anyone can chime in on any subject, including people who don't work at GitLab. The person who's responsible for the work decides how to do it, but they should always take each suggestion seriously and try to respond and explain why it may or may not have been implemented.

Kindness

We value caring for others. Demonstrating we care for people provides an effective framework for challenging directly and delivering feedback. We disagree with companies that say Evaluate People Accurately, Not "Kindly". We're all for accurate assessment, but we think it must be done in a kind way. Give as much positive feedback as you can, and do it in a public way.

Share

There are aspects of GitLab culture, such as extreme transparency, that are unintuitive to outsiders and new team members. Be willing to invest in people and engage in open dialogue. For example, consider making private issues public wherever possible so that we can all learn from the experience.

Everyone can **remind** anyone in the company about our values. If there is a disagreement about the interpretations, the discussion can be escalated to more people within the company without repercussions.

Share problems you run into, ask for help, be forthcoming with information and **speak up**.

Negative is 1-1

Give negative feedback in the smallest setting possible. One-on-one video calls are preferred. If you are unhappy with anything (your duties, your colleague, your boss, your salary, your location, your computer) please let your boss, or the CEO, know as soon as you realize it. We want to solve problems while they are **small**.

Say thanks

Recognize the people that helped you publicly, for example in our #thanks chat channel.

Give feedback effectively

Giving feedback is challenging, but it's important to deliver it effectively. When providing feedback, always make it about the work itself; focus on the business impact and not the person. Make sure to provide at least one clear and recent example. If a person is going through a hard time in their personal life, then take that into account. An example of giving positive feedback is our thanks chat channel. For managers, it's important to realize that employees react to a negative incident with their managers six times more strongly than they

do to a positive one. Keeping that in mind, if an error is so inconsequential that the value gained from providing criticism is low, it might make sense to keep that feedback to yourself. In the situations where negative feedback must be given, focus on the purpose for that feedback: to improve the employee's performance going forward. Give recognition generously, in the open, and often to generate more engagement from your team.

Get to know each other

We use a lot of text-based communication, and if you know the person behind the text, it will be easier to prevent conflicts. So we encourage people to get to know each other on a personal level through our company calls, virtual coffee chats, and during GitLab Contribute.

Don't pull rank

If you have to remind someone of the position you have in the company, you're doing something wrong. People already know our decision-making process. Explain why you're making the decision, and respect everyone irrespective of their function.

Assume positive intent

We naturally have a double standard when it comes to the actions of others. We blame circumstances for our own mistakes, but individuals for theirs. This double standard is called the Fundamental Attribution Error. In order to mitigate this bias you should always assume positive intent in your interactions with others, respecting their expertise and giving them grace in the face of what you might perceive as mistakes.

Address behavior, but don't label people

There is a lot of good in this article about not wanting jerks on our team, but we believe that **jerk** is a label for behavior rather than an inherent classification of a person. We avoid classifications.

Say sorry

If you made a mistake, apologize. Saying sorry is not a sign of weakness but one of strength. The people that do the most work will likely make the most mistakes. Additionally, when we share our mistakes and bring attention to them, others can learn from us, and the same mistake is less likely to be repeated by someone else.

No ego

Don't defend a point to win an argument or double-down on a mistake. You are not your work; you don't have to defend your point. You do have to search for the right answer with help from others.

See others succeed

A candidate who has talked to a lot of people inside GitLab said that, compared to other companies, one thing stood out the most: everyone here mentioned wanting to see each other succeed.

People are not their work

Always make suggestions about examples of work, not the person. Say, "you didn't respond to my feedback about the design" instead of "you never listen". And, when receiving feedback, keep in mind that feedback is the best way to improve and that others want to see you succeed.

Do it yourself

Our collaboration value is about helping each other when we have questions, need critique, or need help. No need to brainstorm, wait for consensus, or do with two what you can do yourself.

Blameless problem solving

Investigate mistakes in a way that focuses on the situational aspects of a failure's mechanism and the decision-making process that led to the failure rather than cast blame on a person or team. We hold blameless root cause analyses and retrospectives for stakeholders to speak up without fear of punishment or retribution.

Short toes

People joining the company frequently say, "I don't want to step on anyone's toes." At GitLab we should be more accepting of people taking initiative in trying to improve things. As companies grow their speed of decision making goes down since there are more people involved. We should counteract that by having short toes and feel comfortable letting others contribute to our domain.

It's impossible to know everything

We know we must rely on others for the expertise they have that we don't. It's OK to admit you don't know something and to ask for help, even if doing so makes you feel vulnerable. It is never too late to ask a question, and by doing so you can get the information you need to produce results and to strengthen your own skills as well as GitLab as a whole. After your question is answered, please document the answer so that it can be shared.

Don't display surprise when people say they don't know something, as it is important that everyone feels comfortable saying "I don't know" and "I don't understand." (As inspired by Recurse.)

Results

We do what we promised to each other, customers, users, and investors.

Dogfooding

- We use our own product. Our development organization uses GitLab.com to manage the DevOps lifecycle of the GitLab.
- Our entire company uses GitLab to collaborate on this handbook. We also capture other content and processes in Git repos and manage them with GitLab.
- When something breaks, doesn't work well, or needs improvement, we are more likely to notice it internally and address it before it impacts our larger community.

Measure results not hours

We care about what you achieve; the code you shipped, the user you made happy, and the team member you helped. Someone who took the afternoon off shouldn't feel like they did something wrong. You don't have to defend how you spend your day. We trust team members to do the right thing instead of having rigid

rules. Do not incite competition by proclaiming how many hours you worked yesterday. If you are working too many hours talk to your manager to discuss solutions.

Give agency

We give people agency to focus on what they think is most beneficial. If a meeting doesn't seem interesting and someone's active participation is not critical to the outcome of the meeting, they can always opt to not attend, or during a video call they can work on other things if they want. Staying in the call may still make sense even if you are working on other tasks, so other peers can ping you and get fast answers when needed. This is particularly useful in multi-purpose meetings where you may be involved for just a few minutes.

Write promises down

Agree in writing on measurable goals. Within the company we use public OKRs for that.

Growth mindset

You don't always get results and this will result in criticism from yourself and/or others. We believe our talents can be developed through hard work, good strategies, and input from others. We try to hire people based on their trajectory, not their pedigree.

Global optimization

This name comes from the quick guide to Stripe's culture. Our definition of global optimization is that you do what is best for the organization as a whole. Don't optimize for the goals of your team when it negatively impacts the goals of other teams, our users, and/or the company. Those goals are also your problem and your job. Keep your team as lean as possible, and help other teams achieve their goals. In the context of collaboration, this means that if anyone is blocked by you, on a question, your approval, or a merge request review, your top priority is always to unblock them, either directly or through helping them find someone else who can, even if this takes time away from your own or your team's priorities.

Tenacity

We refer to this as "persistence of purpose". As talked about in The Influence Blog, tenacity is the ability to display commitment to what you believe in. You keep picking yourself up, dusting yourself off, and quickly get going again having learned a little more.

Ownership

We expect team members to complete tasks that they are assigned. Having a task means you are responsible for anticipating and solving problems. As an owner you are responsible for overcoming challenges, not suppliers, or other team members. Take initiative and pro-actively inform stakeholders when there is something you might not be able to solve.

Sense of urgency

At an exponentially scaling startup time gained or lost has compounding effects. Try to get the results as fast as possible so the compounding of results can begin and we can focus on the next improvement.

Ambitious

While we iterate with small changes, we strive for large, ambitious results.

Perseverance

Working at GitLab will expose you to situations of various levels of difficulty and complexity. This requires focus, and the ability to defer gratification. We value the ability to maintain focus and motivation when work is tough and asking for help when needed.

Bias for Action

It's important that we keep our focus on action, and don't fall into the trap of analysis paralysis or sticking to a slow, quiet path without risk. Decisions should be thoughtful, but delivering fast results requires the fearless acceptance of occasionally making mistakes; our bias for action also allows us to course correct quickly.

Accepting Uncertainty

The ability to accept that there are things that we don't know about the work we're trying to do, and that the best way to drive out that uncertainty is not by layering analysis and conjecture over it, but rather accepting it and moving forward, driving it out as we go along. Wrong solutions can be fixed, but non-existent ones aren't adjustable at all. The Clever PM Blog

Customer results

Our focus is to improve the results that customers achieve, which requires being aware of the Concur effect, see the Hacker News discussion for a specific UX example. **Customer results are more important**than:

1. **What we plan to make**. If we focus only on our own plans, we would have only GitLab.com and no self-hosted delivery of GitLab.
2. **Large customers**. This leads to the innovator's dilemma, so we should also focus on small customers and future customers (users).
3. **What customers ask for**. This means, we don't use the phrase "customer focus" because it tempts us to prioritize what the customer *says* they want over what we discover they actually need through our product development process. Often, it's easier for a customer to think in terms of a specific solution than to think about the core problem that needs to be solved. But a solution that works well for one customer isn't always relevant to other customers, and it may not align with our overall product strategy. When a customer asks for something specific, we should strive to understand why, work to understand the broader impact, and then create a solution that scales.
4. **Our existing scope**. For example, when customers asked for better integrations and complained about integration costs and effort, we responded by expanding our scope to create a single application for the DevOps lifecycle.
5. **Our assumptions**. Every company works differently, so we can't assume that what works well for us will support our customers' needs. When we have an idea, we must directly validate our assumptions with customers to ensure we create scalable, highly relevant solutions.
6. **What we control**. We should take responsibility for what the **customer experiences**, even when it isn't entirely in our control. For example, we looked only at GitLab.com when a customer managed instance downtime can be a $1m a day problem.

Efficiency

We care about working on the right things, not doing more than needed, and not duplicating work. This enables us to achieve more progress, which makes our work more fulfilling.

Write things down

We document everything: in the handbook, in meeting notes, in issues. We do that because "the faintest pencil is better than the sharpest memory." It is far more efficient to read a document at your convenience than to have to ask and explain. Having something in version control also lets everyone contribute suggestions to improve it.

Boring solutions

Use the simplest and most boring solution for a problem, and remember that "boring" should not be conflated with "bad" or "technical debt." The speed of innovation for our organization and product is constrained by the total complexity we have added so far, so every little reduction in complexity helps. Don't pick an interesting technology just to make your work more fun; using established, popular tech will ensure a more stable and more familiar experience for you and other contributors.

Make a conscious effort to **recognize** the constraints of others within the team. For example, sales is hard because you are dependent on another organization, and development is hard because you have to preserve the ability to quickly improve the product in the future.

Efficiency for the right group

Optimize solutions globally for the broader GitLab community. Making a process efficient for one person or a small group may not be the efficient outcome for the whole GitLab community. As an example, it may be best to choose a process making things slightly less efficient for you while making things massively more efficient for thousands of customers. For example: choose to replace a renewal process that requires 1,000s of customers to spend 2 hours each with one that only takes 60 seconds even when it may make a monthly report less efficient internally! In a decision, ask yourself "for whom does this need to be most efficient?". Quite often, the answer may be your users, contributors, customers, or team members that are dependent upon your decision.

Be respectful of others' time

Consider the time investment you are asking others to make with meetings and a permission process. Try to avoid meetings, and if one is necessary, try to make attendance optional for as many people as possible. Any meeting should have an agenda linked from the invite, and you should document the outcome. Instead of having people ask permission, trust their judgment and offer a consultation process if they have questions.

Spend company money like it's your own

Every dollar we spend will have to be earned back; be as frugal with company money as you are with your own.

Frugality

Amazon states it best with: "Accomplish more with less. Constraints breed resourcefulness, self-sufficiency and invention. There are no extra points for growing headcount, budget size or fixed expense."

ConvDev

We work according to the principles of conversational development.

Freedom

You should have clear objectives and the freedom to work on them as you see fit.

Short verbal answers

Give short answers to verbal questions so the other party has the opportunity to ask more or move on.

Keep broadcasts short

And keep one-to-many written communication short, as mentioned in this HBR study: "A majority say that what they read is frequently ineffective because it's too long, poorly organized, unclear, filled with jargon, and imprecise."

Managers of one

We want each team member to be a manager of one who doesn't need daily check-ins to achieve their goals.

Responsibility over rigidity

When possible we give people the responsibility to make a decision and hold them accountable for that instead of imposing rules and approval processes.

Accept mistakes

Not every problem should lead to a new process to prevent them. Additional processes make all actions more inefficient, a mistake only affects one.

Move fast by shipping the minimum viable change

We value constant improvement by iterating quickly, month after month. If a task is too big to deliver in one month, cut the scope.

Diversity & Inclusion

Diversity and inclusion are fundamental to the success of GitLab. We aim to make a significant impact in our efforts to foster an environment where everyone can thrive. We are designing a multidimensional approach to ensure that GitLab is a place where people from every background and circumstance feel like they belong and can contribute. We actively chose to build and institutionalize a culture that is inclusiveand supports all employees equally in the process of achieving their professional goals. We hire globally and encourage hiring in a diverse set of countries. We work to make everyone feel welcome and to increase the participation of underrepresented minorities and nationalities in our community and company. For example our sponsorship of diversity events and a double referral bonus.

Bias towards asynchronous communication

Take initiative to operate asynchronously whenever possible. This shows care and consideration for those who may not be in the same time zone, are traveling outside of their usual time zone, or are structuring their day around pressing commitments at home or in their community.

This is demonstrated by communicating recordings of meetings, using GitLab Issues and Merge Requests rather than texts, calls, or Slack messages, and being sensitive to local holidays and vacation statuses. Encourage others to default to documentation rather than pressuring others to be online outside of their working hours.

Reach across company departments

While it's wise to seek advice from experts within your function, we encourage GitLab team members to seek and provide feedback across departments. This enables the team to iterate more quickly, taking a more diverse perspective into account.

Make family feel welcome

One of the unique elements to an all-remote culture is the ability to visit a person's home while collaborating. If the tenor of the meeting allows, feel welcome to invite your family members or pets to drop by and greet your colleagues.

Shift working hours for a cause

Caregiving, outreach programs, and community service do not conveniently wait for regular business hours to conclude. If there's a cause or community effort taking place, feel welcome to work with your manager and shift your working hours to be available during a period where you'll have the greatest impact for good. For colleagues supporting others during these causes, document everything and strive to post recordings so it's easy for them to catch up.

Be a mentor

People feel more included when they're supported. To encourage this, and to support diversified learning across departments, consider GitLab's Interning for Learning program.

Culture fit is a bad excuse

We don't hire based on culture or select candidates because we'd like to have a drink with them. We hire and reward employees based on our shared values as detailed on this page. We want a **values fit**, not a culture fit. We want cultural diversity instead of cultural conformity, for example, a brogrammeratmosphere. Said differently: "culture add" > "culture fit" or "hire for culture contribution" since our mission is everyone can contribute.

Religion and politics at work

We generally avoid discussing politics or religion in public forums because it is easy to alienate people that have a minority opinion. This doesn’t mean we never discuss these topics. Because we value diversity and inclusion, and want all team members to feel welcome and contribute equally, we encourage free discussion of operational decisions that can move us toward being a more inclusive company. GitLab also publicly supports pro-diversity activities and events.

It's also acceptable for individuals to bring up politics and religion in social contexts such as coffee chats and real-life meetups with other coworkers (with the goal to understand and not judge), but always be aware of sensitivities, exercise your best judgment, and make sure you stay within the boundaries of our Code of Conduct.

Quirkiness

Unexpected and unconventional things make life more interesting. Celebrate and encourage quirky gifts, habits, behavior, and points of view. An example is our company call where we spend most of our time talking about what we did in our private lives, from fire-throwing to knitting. Open source is a great way to interact with interesting people. We try to hire people who think work is a great way to express themselves.

Building a safe community

Do **not** make jokes or unfriendly remarks about race, ethnic origin, skin color, gender, or sexual orientation. Everyone has the right to feel safe when working for GitLab and/or as a part of the GitLab community. We do not tolerate abuse, harassment, exclusion, discrimination or retaliation by/of any community members, including our employees. You can always **refuse** to deal with people who treat you badly and get out of situations that make you feel uncomfortable.

Unconscious bias

We recognize that unconscious bias is something that affects everyone and that the effect it has on us as humans and our company is large. We are responsible for understanding our own implicit biases and helping others understand theirs. We are continuously working on getting better at this topic.

Inclusive benefits

We list our Transgender Medical Services and Parental Leave publicly so people don't have to ask for them during interviews.

Inclusive language & pronouns

Use **inclusive** language. For example, prefer "Hi everybody" or "Hi people" to "Hi guys". While there are several good guides from folks like 18f, University of Calgary, and Buffer on using inclusive language, we don't keep an exhaustive list. When new possibly non-inclusive words arise, we prefer to be proactive and look for an alternative. If your goal is to be inclusive, it is more effective to make a small adjustment in the vocabulary when some people have a problem with it, rather than making a decision to not change it because some people don't think it is a problem. And if you make a mistake (e.g., accidentally using the wrong pronoun or an outdated phrase), acknowledge it, **apologize gracefully and move on**, no need to dwell on it at the moment. After, you can work hard to avoid making that mistake in the future. Please also visit our Gender and Sexual-orientation Identity Definitions and FAQ page if you have questions around pronouns and other gender/sexual-orientation related topics.

Inclusive interviewing

This is documented on our page about interviewing.

See Something, Say Something

As a globally dispersed company, we have employees from many different backgrounds and cultures. That means it is important for each of us to use great judgment in being respectful and inclusive of our teammates. At the same time, we may sometimes not fully realize we have said or done something to offend someone. It is important that our teammates hold each other accountable and let them know if they have unintentionally or intentionally done something so they can learn and gain additional understanding of perspectives different from our own. It is also important that our teammates don't feel excluded or minimized

by the words we use or the things we do. Thus, we all need to speak up when we see something that isn't respectful or inclusive.

Neurodiversity

Neurodiversity is a type of diversity that includes Autism, ADHD, and other styles of neurodivergent functioning. While they often bring unique skills and abilities, which can be harnessed for competitive advantage in many fields including cybersecurity, neurodivergent individuals are often discriminated against, and sometimes have trouble making it through traditional hiring processes. These individuals should be able to contribute as GitLab team-members. The handbook, values, strategy, and interviewing process should never discriminate against the neurodivergent.

Family and friends first, work second

Long lasting relationships are the rocks of life and come before work. As someone said in our #thanks channel, after helping a family member for 5 days after a hurricane: "THANK YOU to GitLab for providing a culture where "family first" is truly meant".

Iteration

We do the smallest thing possible and get it out as quickly as possible. If you make suggestions that can be excluded from the first iteration turn them into a separate issue that you link. Don't write a large plan, only write the first step. Trust that you'll know better how to proceed after something is released. You're doing it right if you're slightly embarrassed by the minimal feature set shipped in the first iteration. This value is the one people most underestimate when they join GitLab. The impact both on your work process and on how much you achieve is greater than anticipated. In the beginning it hurts to make decisions fast and to see that things are changed with less consultation. But frequently the simplest version turns out to be the best one.

People that join GitLab all say they already practice this iteration. But this is the value that they have the hardest time adopting. People are trained that if you don't deliver a perfect or polished thing you get dinged for it. If you do just one piece of something you have to come back to it. Doing the whole thing seems more efficient, even though it isn't. If the complete picture is not clear your work might not be perceived as you want it to be perceived. It seems better to make a comprehensive product. They see other people in the GitLab organization being really effective with iteration but don't know how to make the transition, and it's hard to shake the fear that constant iteration can lead to shipping lower-quality work or a worse product.

The way to resolve this is to write down only what you can do with the time you have for this project right now. That might be 5 minutes or 2 hours. Think of what you can complete in that time that would improve the current situation. Iteration can be uncomfortable, even painful. If you're doing iteration correctly, it should be.

However, if we take smaller steps and ship smaller simpler features, we get feedback sooner. Instead of spending time working on the wrong feature or going in the wrong direction, we can ship the smallest product, receive fast feedback, and course correct. People might ask why something was not perfect. In that case, mention that it was an iteration, you spent only "x" amount of time on it, and that the next iteration will contain "y" and be ready on "z".

Don't wait

Don’t wait. When you have something of value like a potential blog post or a small fix implement it straight away. Right now everything is fresh in your head and you have the motivation, inspiration is perishable. Don’t wait until you have a better version. Don’t wait until you record a better video. Don’t wait for an event (like

Contribute). Inventory that isn’t released is a liability since it has to be managed, gets outdated, and you miss out on the feedback you get when you would have implemented it straight away.

Reduce cycle time

Short iterations reduce our cycle time.

Work as part of the community

Small iterations make it easier to work with the wider community. Their work looks more like our work, and our work is also quicker to receive feedback.

Minimum Viable Change (MVC)

We encourage MVCs to be as small as possible. Always look to make the quickest change possible to improve the user's outcome. If you validate that the change adds more value than what is there now, then do it. No need to wait for something more robust. More information is in the product handbook, but this applies to everything we do in all functions. Specifically for product MVCs, there is additional responsibility to validate with customers that we're adding useful functionality without obvious bugs or usability issues.

Make a proposal

If you need to decide something as a team, make a concrete proposal instead of calling a meeting to get everyone's input. Having a proposal will be a much more effective use of everyone's time. Every meeting should be a review of a proposal. We should be brainwriting on our own instead of brainstorming out loud. State the underlying problem so that people have enough context to propose reasonable alternatives. The people that receive the proposal should not feel left out and the person making it should not feel bad if a completely different proposal is implemented. Don't let your desire to be involved early or to see your solution implemented stand in the way of getting to the best outcome. If you don't have a proposal don't let that stop you from highlighting a problem, please state that you couldn't think of a good solution and list any solutions you considered.

Everything is in draft

At GitLab we rarely put draft on any content or proposals. Everything is always in draft and subject to change.

Under construction

As we get more users they will ask for stability, especially in our UX. We should always optimize for the long term. This means that users will be inconvenienced in the short term, but current and future users will enjoy a better product in the end.

Low level of shame

When we talked to Nat Friedman he said: "A low level of shame is intrinsic to your culture.". This captures the pain we feel by shipping something that isn't where we want it to be yet.

Focus on Improvement

We believe great companies sound negative because they focus on what they can improve, not on what is working. Our first question in every conversation with someone outside the company should be: what do you think we can improve? This doesn't mean we don't recognize our successes, for example see our Say Thanks value. We are positive about the future of the company; we are present day pessimists and long term optimists.

Do things that don't scale

First optimize for speed and results; when it is a success figure out how to scale it. Great examples are in this article by Paul Graham.

Make two-way door decisions

Most decisions are easy to reverse, have the directly responsible individual make them without approval. As Jeff Bezos describes only when you can't reverse them should there be a more thorough discussion.

Changing proposals isn't iteration

Changing a proposal isn't iterating. Only when the change is rolled out to users can you learn from feedback. When you're changing a proposal based on different opinions you're frequently wasting time; it would be better to roll out a small change quickly and get real world feedback.

A few challenges have arisen with how we approach iteration. The best example may be the proposal of a 2-month release cycle. The argument was that a longer release cycle would buy us time for bug fixes and feature development, but we don’t believe that is the case. As detailed above, we aim to make the absolute smallest thing possible and that doing otherwise will only slow us down.

That said, we would love to work on a 2-week release cycle, but that should be another conversation.

Make small merge requests

When you are submitting a merge request for a code change, or a process change in the handbook, keep it as small as possible. If you are adding a new page to the handbook, create the new page with a small amount of initial content, get it merged quickly, and then add additional sections iteratively with subsequent merge requests. Similarly, if you are adding a large feature to the software, create small, iterative merge requests for the different aspects. These merge requests might not provide immediate value, as long as they do not break anything, and have appropriate tests for new functionality. If you aren't sure how to split something into small, iterative merge requests, consider kindly asking your team and/or maintainers for suggestions. If you are asked to review a merge request that is too big, consider kindly asking the author to split it into smaller merge requests before reviewing.

When we iterate slowly

In some cases, rapid iteration can get in the way of results. For example when adjusting our marketing messaging (where consistency is key), product categories (where we've set development plans), sales methodologies (where we've trained our teams) and this values page (where we use the values to guide all GitLab team-members). In those instances we add additional review to the approval process, not to prohibit, but to be more deliberate in our iteration. The change process is documented on the page and takes place via merge request approvals.

Transparency

Be open about as many things as possible. By making information public we can reduce the threshold to contribution and make collaboration easier. Use public issue trackers, projects, and repositories when possible.

An example is the public repository of this website that also contains this company handbook. Everything we do is public by default, for example, the GitLab CE and GitLab EE issue trackers, but also marketingand infrastructure. Transparency creates awareness for GitLab, allows us to recruit people that care about our values, it gets us more and faster feedback from people outside the company, and makes it easier to collaborate with them. It is also about sharing great software, documentation, examples, lessons, and processes with the **whole community** and the world in the spirit of open source, which we believe creates more value than it captures.

There are exceptions. Material that is not public by default is documented. We are above average at keeping things confidential that need to be. On a personal level, you should tell it like it is instead of putting up a poker face. Don't be afraid to admit you made a mistake or were wrong. When something went wrong it is a great opportunity to say "What's the kaizen moment here?" and find a better way without hurt feelings.

Even as we move towards becoming a public company and beyond, we know that our value of transparency will be key to our success. Often company values get diluted as they grow, most likely because they do not write anything down. But we will make sure our values scale with the company. When we go public, we can declare everyone in the company as an insider, which will allow us to remain transparent internally about our numbers, etc. Everything else that can be transparent will continue to be so.

Public by default

Everything at GitLab is public by default. If something is not public there should be a reference in the handbook that states a confidential decision was taken with a link to our Not Public guidelines unless legal feels it carries undue risk. The public process does two things: allows others to benefit from the conversation and acts as a filter; since there is only a limited amount of time, we prioritize conversations that a wider audience can benefit from.

Not Public

Most things are public by default. However there are some things that are not public because of the sensitivity of the issue or requirements.

Directness

Being direct is about being transparent with each other. We try to channel our inner Ben Horowitz by being both straightforward and kind, an uncommon cocktail of no-bullshit and no-asshole. Feedback is always about your work and not your person. That doesn't mean it will be easy to give nor receive it.

Surface issues constructively

Be transparent to the right people (up) at the right time (when still actionable). If you make a mistake, don't worry; correct it and **proactively** let the affected party, your team, and the CEO know what happened, how you corrected it and how, if needed, you changed the process to prevent future mistakes.

Anyone and anything can be questioned

Any past decisions and guidelines are open to questioning as long as you act in accordance with them until they are changed.

Disagree, commit, and disagree

Everything can be questioned but as long as a decision is in place we expect people to commit to executing it, which is a common principle. Every decision can be changed, our best decision was one that changed an earlier one. When you want to reopen the conversation on something, show that your argument is informed by previous conversations. You have to achieve results on every decision while it stands, even when you are trying to have it changed. You should communicate with the person who can change the decision instead of someone who can't.

Transparency is only a value if you do it when it is hard

For example, many companies in California do not give you the real reason why they declined your application because it increases the chance of legal action. We want to only reject people for the right reasons and we want to give them the opportunity to grow by getting this feedback. Therefore, we'll accept the increased risk of holding ourselves to a high standard of making decisions and do the right thing by telling them what we thought. Other examples are being transparent about security incidents, and participating in and contributing to Live Broadcasts.

Single Source of Truth

By having most company communications and work artifacts be internet-public, we have one single source of truth for all GitLab team-members, users, customers, and other community members. We don‘t need separate artifacts with different permissions for different people.

Find the Limit

We accept that we occasionally make mistakes in the direction of transparency. In other words, we accept it if we're sometimes publicizing information that should have remained confidential in retrospect. Most companies become non-transparent over time because they don't accept any mistakes. Making mistakes and reflecting on them means we know where the limit of transparency is.

Say why, not just what

Transparent changes have the reasons for the change laid out clearly along with the change itself. This leads to fewer questions later on because people already have some understanding. A change with no public explanation can lead to a lot of extra rounds of questioning which is less efficient. Avoid using terms such as "industry standard" or "best practices" as they are vague, opaque, and don't provide enough context as a reason for a change.

Reproducibility

Enable everybody involved to come to the same conclusion as you. This not only involves reasoning, but also, for example providing raw data and not just plots; scripts to automate tasks and not just the work they have done; and documenting steps while analyzing a problem. Do your best to make the line of thinking transparent to others even if they may disagree.

Accountability

Increases accountability when making decisions and difficult choices.

Five dysfunctions

Our values help us to prevent the five dysfunctions.

1. **Absence of trust** Unwilling to be vulnerable within the group => *prevented by collaboration, specifically* kindness
2. **Fear of conflict** Seeking artificial harmony over constructive passionate debate => *prevented by transparency, specifically* directness *and collaboration, specifically* short toes
3. **Lack of commitment** Feigning buy-in for group decisions creates ambiguity throughout the organization => *prevented by transparency, specifically* directness
4. **Avoidance of accountability** Ducking the responsibility to call peers on counterproductive behavior which sets low standards => *prevented by results, iteration, and* transparency
5. **Inattention to results** Focusing on personal success, status, and ego before team success => *prevented by* results

Some dysfunctions are not addressed directly by our values, for example trust is not one of our values. Similar to happiness, trust is something that is an outcome, not something you can strive for directly. We hope that the way we work and our values will instill trust, instead of mandating it from people; trust is earned, not given.

Why have values

Our values should give guidelines on how to behave and must be actionable. They help us describe the type of behavior that we expect from people we hire. They help us to know how to behave in the organization and what to expect from others. Values are a framework for distributed decision making; they allow you to determine what to do without asking your manager.

Hierarchy

Occasionally, values can contradict each other. For instance, transparency would dictate we publish all security vulnerabilities the moment they are found, but this would jeopardize our users. It's useful to keep in mind this hierarchy to resolve confusion about what to do in a specific circumstance, while remaining consistent with our core values.

<table>
<tr><td></td><td>Results</td><td></td><td>It is most important to focus on results.</td></tr>
<tr><td></td><td>Iteration</td><td>Transparency</td><td>We trust these values will lead to results.</td></tr>
</table>

Collaboration	**D**iversity & Inclusion	**E**fficiency	Distinguish us from other companies.

Updating our values

Our values are updated frequently and as needed. Everyone is welcome to make a suggestion to improve them. To update: make a merge request and assign it to the CEO. If you're a team member or in the core team please post a link to the MR in the #values channel. If you're not part of those groups please send a direct Twitter message to @sytses.

How do we reinforce our values

Whatever behavior you reward will become your values. We reinforce our values by what:

1. Leadership does.
2. We select for during hiring.
3. We emphasize during on-boarding.
4. Behavior we give each-other 360 feedback on.
5. Behavior we compliment.
6. Criteria we use for discretionary bonuses.
7. Criteria we use for our annual compensation review.
8. Criteria we use for promotions.
9. Criteria we use to manage underperformance.
10. We do when we let people go.
11. We give value awards for during Contribute.

In negative feedback we should be specific about what the problem is. For example, saying someone is "not living the values" isn't helpful.

Permission to play

From our values we excluded some behaviors that are obvious, we call them our permission to play behavior:

- Be truthful and honest.
- Be dependable, reliable, fair, and respectful.
- Be committed, creative, inspiring, and passionate.
- Be deserving of the trust of our users and customers.
- Act in the best interest of the company, our team members, our customers, users, and investors.
- Act in accordance with the law.
- Don't show favoritism as it breeds resentment, destroys employee morale, and creates disincentives for good performance. Seek out ways to be fair to everyone.

What is not a value

- All remote isn't a value. It is something we do because it helps to practice our values of transparency, efficiency, diversity & inclusion, and results.

Question from new team members

During every GitLab 101 session with new hires we discuss our values. We document the questions and answers to Frequently Asked Questions about the GitLab Culture.

Mission

Our mission is that **everyone can contribute**. This mission guides our path, and we live our values along that path.

Biggest risks

We have a page which documents our biggest risks, many of our values help to mitigate some of these risks.

https://about.gitlab.com/company/

GitLab is an open core company which develops software for the software development lifecycle used by more than 100,000 organizations and has an active community of more than 2200 contributors. GitLab openly shares more information than most companies and is public by default, meaning our projects, strategy, direction and metrics are discussed openly and can be found within our website. Our values are Collaboration, Results, Efficiency, Diversity & Inclusion, Iteration, and Transparency (CREDIT) and these form our culture.

GitLab's team handbook, if printed would be over 3,000 pages of text, is the central repository for how we operate and is a foundational piece to the GitLabvalues.

GitLab believes in a world where everyone can contribute. Our mission is to change all creative work from read-only to read-write. When everyone can contribute, consumers become contributors and we greatly increase the rate of human progress.

With GitLab, everyone can contribute.

We release every month on the 22nd and there is a publicly viewable direction for the product.

Learn more from our blog

GitLab Inc. is an open-core company that sells subscriptionsthat offer more features and support for GitLab.

Learn about open core

GitLab, the product is a complete DevOps platform,delivered as a single application, fundamentally changing the way Development, Security, and Ops teams collaborate.

Learn more about our product

All remote with

1032

team members

Located in

64

countries

GitLab Inc. is an active participant in this community, see our stewardship for more information.

See our stewardship

2011

GitLab, the open source project began.

2015

We joined Y Combinator and started growing faster.
Join our team.

Most of our internal procedures can be found in a publicly viewable 2000+ page handbook and our objectives are documented in our OKRs.

Our mission is to change all creative work from read-only to read-write so that **everyone can contribute.** This is part of our overall strategy.

Our values are Collaboration, Results, Efficiency, Diversity & Inclusion, Iteration, and Transparency (CREDIT) and these form an important part of our culture.

Our Tanuki (Japanese for raccoon dog) logo symbolizes this with a smart animal that works in a group to achieve a common goal, you can download it on our press page.

https://about.gitlab.com/company/culture/

Introduction

Please see our company page for more general information about GitLab. You can see how our team has grown at the GitLab Contribute page.

It's an exciting time to be part of GitLab. We're a fast-growing, all-remote team, and we're looking for people to join us around the world. Here's a look at what you can expect from our culture and all-remote environment.

Everyone can contribute

Our size and our mission (that everyone can contribute) mean that our team members can — and are expected to — make an impact across the company.

Because we all use our product internally, you don't have to be a developer to learn to collaborate in the GitLab tool. From your very first week, no matter your role, you'll gain the technical skills needed to access, question, and contribute to projects far beyond your job description.

This unique approach works because we're a team of helpful, passionate people who want to see each other, the company, and the broader GitLab community succeed. We learn from each other, challenge each other, and thank each other.

Come prepared to do meaningful work that will help shape the future of the company.

While the opportunities to contribute are boundless in a growing organization like GitLab, they may not be clearly defined. You'll need to think creatively, speak up to see how you can help, and be willing to try something new.

Freedom to iterate

At GitLab, our value of iteration has a unique impact on the way we operate and get things done.

Working this way means our team members are expected to quickly deliver the minimum viable change in their work instead of waiting to produce a polished, completed product.

While this can be a challenging practice to adopt at first, it's liberating to be able to make mistakes, get feedback quickly, and course correct to reach a better outcome, faster.

As our company and the industry continue to grow, you'll have the freedom to change and constantly evolve everything from your schedule and your workspace to your job description and your skills.

All-remote work

Being part of our all-remote team offers unique advantages beyond the requisite flexibility you'll find in many organizations.

As a GitLab team member, you can work from anywhere with good internet. Whether you're an adventurer looking to travel the world while still pursuing your career, a parent or caregiver who wants a job that allows you to spend more time with family, or somewhere in between, you'll have the freedom to contribute when and where you do your best work.

But there's more to our all-remote culture than the daily flexibility it provides. By nature, having no offices or headquarters makes us more inclusive, more transparent, and more efficient in everything we do. With a team spread across over 60 countries around the globe, we invite diverse perspectives, we document everything, and we collaborate asynchronously.

Despite all of its benefits for team members, our company, and the world, remote work isn't for everyone. Learn more about all-remote work at GitLab and decide if it's right for you.

Advantages

Top 10 reasons to work for GitLab:

1. Work with helpful, kind, motivated, and talented people.
2. Work remote so you have no commute and are free to travel and move.
3. Have flexible work hours so you are there for other people and free to plan the day how you like.
4. Everyone works remote, but you don't *feel* remote. We don't have a head office, so you're not in a satellite office.
5. Work on open source software so you can interact with a large community and can show your work.

6. Work on a product you use every day: we drink our own wine.
7. Work on a product used by lots of people that care about what you do.
8. As a company we contribute more than we take, most of our work is released as the open source GitLab CE.
9. Focused on results, not on long hours, so that you can have a life and don't burn out.
10. Open internal processes: know what you're getting in to and be assured we're thoughtful and effective.

Other pages related to culture

- GitLab 101
- GitLab Contribute
- Internal Feedback
- Diversity and Inclusion
- Top Team Member
- All Remote

Historical Anecdotes

October 8th, 2011

Dmitriy started GitLab when he pushed the initial commit.

August 24th, 2012 ***Sid announced GitLab on HN.***

September 14th, 2012 ***First 10 people get access to GitLab Cloud (now known as GitLab.com).***

November 13th, 2012 ***GitLab CI is officially announced.***

July 22nd, 2013 ***GitLab Enterprise Edition is announced.***

April 18th, 2014 ***GitLab Cloud renamed to GitLab.com.***

March 4th, 2015 ***GitLab in Y Combinator winter 2015 batch.***

August 15th, 2015 ***Series A Funding was signed.***

October 10th, 2015 ***Anniversary of our first ever summit in Amsterdam with 25 GitLab team-members.***

Team Stories What better way to convey a sense of who we are and how we work together, than by sharing the stories about it?

The Boat

Back then, the whole team used to fit in one car. And the car was called "the Boat".

We even took the Boat from San Francisco to Las Vegas to celebrate Job's bachelor party, but as you can see in this video, he thought we were going to visit a customer in Los Angeles!

The cattle

Staring down the cattle?

Our CFO, Paul, was on vacation on a cattle ranch, during a time of fundraising. Normally vacation is vacation of course, but in this case it was necessary to have some calls now and again which required strong internet. To get to strong internet, Paul had to cross fields with cattle in them, and stare them down. Over the course of many trips he learned that cattle are docile, mostly... but don't turn your back on them because they can't be outrun!

IPO date comes in handy… 2 years out

After spending a couple of days in meetings with customers in New York City, USA, Sid and Kirsten had a few hours before their flight and wanted to visit the WTC Observatory deck. It didn't work out but our IPO date did work out in their favor. In the keynote at our Cape Town event, Sid explains what happened.

Being new to GitLab, our CRO, Michael McBride joined Sid in meeting with customers in New York City, USA where customers got a glimpse of what it's like to work at GitLab for him

Planned date of November 18th, 2020 to take GitLab public

Many people ask "why are you going public on November 18th, 2020?" November 18th 2020 was set for the following reasons:

1. It is roughly 10 years after DZ started working on the project
2. It is roughly 5 years after the first employees received 4 year stock option vesting schedules
3. When determined to do a public offering in 2020, it was as late as possible in 2020, as markets do not typically move in December
4. It was originally decided to be November 16th, since that was the last week before Thanksgiving. Paul (our CFO) knew this because it was the birthday of his twins.
5. Minutes after publishing the date Paul looked at the calendar and saw it was a Monday. This isn't a great day of the week to go public since people are digesting the news of the weekend, so in 20 minutes after publishing the date was moved Wednesday, November 18th, 2020.
6. Only after picking the new date Sid realized that it would have been the 100th birthday of the grandfather he is named after.

https://about.gitlab.com/blog/2015/04/08/the-remote-manifesto/
We all have been greatly helped by Scrum and the Agile manifesto. It freed us from waterfall planning and excessive process. But working remotely and continuous delivery need something more. At GitLab we love to work remotely, but that means we need to communicate as effectively as possible.The following are GitLab's eight principles for modern teams working remotely:

1. Work from anywhere you want

Working remotely allows you to be there for the ones you love, and be more available for them. It allows you to see more places, without ever having to commute. On top of that, working remotely removes almost every distraction.

2. Communicate Asynchronously

Don't try to mimic an office. Communicate using issue mentions and chat tools. Reduce task switching and put an end to email overload. Choose the right channel of communication according to the necessity of the task you're working on. Can it wait a few minutes, a few hours, even a few days? Don't take someone from their work if you don't have to. If people *are* working from the same location, it is important that they do not skimp on writing things down. Everyone should use the same tools to communicate.

3. Recognize that the future is unknown

Ship stuff when it's done, not when the sprint (planning) is complete.

4. Have face-to-face meetings online

There is no need to cut back on face-to-face meetings. The technology is readily available and it's easier to use than ever. We're human, we like to converse. Some times it can be critical to talk, even if only for a minute, when all other communication is written.

5. Daily stand-up meetings are for bonding, blockers and the future

Don't talk about what you did yesterday, this is not a reporting moment where everyone tries to look busy. Rather, kickstart the day with some bonding, solve anything blocking and share future plans so people can plan and act and ultimately save time.

6. Bond in real life

Hanging out together in real life is awesome and totally worth it. These are the best days of our lives. Spend time together and make sure to do more than just work. Do a martial arts workshop together, visit the parents of an employee, go to a festival together: have fun.

7. Give credit where it's due and remember to say thank you

At GitLab we have a Slack channel `#thanks` for this purpose. It always feels good to give and receive a thanks