









GitLab Values

Learn more about how we live our values at GitLab

CREDIT

GitLab's six core values are  [Collaboration](#),  [Results for Customers](#),  [Efficiency](#),  [Diversity, Inclusion & Belonging](#),  [Iteration](#), and  [Transparency](#), and together they spell the **CREDIT** we give each other by assuming good intent. We react to them [with values emoji](#) and they are made actionable below.

About our values



We take inspiration from other companies, and we always go for the [boring solutions](#). Our co-founder, Sid Sijbrandij, has [shared the origin](#) of each of the CREDIT values, but just like the rest of our work, we continually adjust our values and strive to make them better. GitLab values are a living document. In many instances, they have been documented, refined, and revised based on lessons learned (and scars earned) in the course of doing business.

We used to have more values, but it was difficult to remember them all. In response, we condensed them, created an acronym (CREDIT), and listed operating principles to guide behavior.

Everyone is welcome to suggest improvements. Please assign MRs to update these values to our [Chief People Officer](#) and if you work at GitLab, also @mention them in the [#values Slack channel](#).



Driving Results with CREDIT

GitLab

03:42

[Driving Results with CREDIT](#) from [GitLab](#) on [Vimeo](#).

Collaboration

To achieve results, team members must work together effectively. At GitLab, 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. Kindness doesn't mean holding back on feedback or avoiding disagreements, these are crucial to professional growth and getting results for customers. Kindness means you make a separation between the work and the person, you can criticize someone's work but still be respectful to the person. Give as much positive feedback as you can, and do it in a public way.

Share

There are aspects of GitLab culture, such as intentional 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. Don't be afraid of judgement or scrutiny when sharing publicly, we all understand [it's impossible to know everything](#).

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 feedback is 1-1

Give negative feedback in the smallest setting possible. One-on-one video calls are preferred.

Negative *feedback* is distinct from negativity and disagreement. If there is no direct feedback involved, strive to discuss disagreement [in a public channel](#), respectfully and [transparently](#).

In a [GitLab Unfiltered interview on values](#), GitLab co-founder Sid Sijbrandij offers the following context.

We deal with negative all the time at GitLab. If it's not a problem, then why are we discussing it? We deal with negativity a lot, and that's also part of our ambition.

If you want to get better, you talk about what you can improve. We're allowed to publicly discuss negative things; we're not allowed to give negative feedback in a large setting if it could be feasibly administered in a smaller setting.

Negative feedback can be given in a group setting if it's to someone higher in the management chain. This shows that no one is above feedback.

Provide feedback in a timely manner

We want to solve problems while they are **small**. If you are unhappy with anything (your duties, your colleague, your boss, your salary, your location, your computer), please voice your concerns rather than keeping them to yourself. If you need to escalate beyond your manager, you could consider speaking to your [skip-level](#), a more senior person, or a [people business partner](#).

Say thanks

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

When publicly thanking, it's important to recognize the following:

- Showing thanks in as large a setting as possible (company-wide) at a company as large as ours is the exception instead of the norm, it takes some getting used to.
- Being thanked at the company level for what you view as a relatively small or minuscule contribution can feel awkward.
- Thanking a person in [#thanks](#) should be done sincerely and summarize why you are thankful so the person on the receiving end can easily understand why they are being thanked. Even while [assuming positive intent](#), not all folks are comfortable with public praise. Help this person understand how they went above and beyond and why you felt it was important for the team member to be recognized.
- There are a number of good ways and places to say thanks. We shouldn't limit saying thanks to just the [#thanks 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 team members 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 team member'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 [informal communication](#), for example, virtual [coffee chats](#), and during [GitLab Summit](#).

Reach across company departments

While it's wise to seek advice from experts within your function, we encourage GitLab team members to do the same across departments. This enables the company to iterate more quickly, embrace the understanding that everyone can contribute and include more diverse perspectives when possible.

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. This includes using the rank of another person - [including the CEO](#) - to sell an idea or decision.

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.

When [disagreeing](#), folks sometimes argue against the weakest points of an argument, or an imaginary argument (e.g. ["straw man"](#)). Assume the points are presented in good faith, and instead try to argue against the strongest version of your opponent's position. We call this arguing against a "steel" position, instead of a "straw" one. This concept is borrowed from [argue the "steel man"](#) technique.

A "steel" position should be against the absolute most effective version of your opponent's position — potentially even more compelling than the one they presented. A good "steel" position is one where the other person feels you've represented their position well, even if they still disagree with your assumptions or conclusion.

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 as soon as possible. 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. Mistakes can include when you have not been kind to someone. In order to reinforce our values, it is important, and takes more courage, to apologize publicly when you have been unkind publicly (e.g., when you have said something unkind or unprofessional to an individual or group in a Slack channel).

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.

In a GitLab Unfiltered [interview](#), GitLab Head of Remote Darren M. adds context on this operating principle.

In many organizations, there's a subtle, low-level, persistent pressure to continually prove your worth. And I believe that this fuels imposter syndrome and wreaks havoc on [mental health](#).

What's so troubling to me is how often perception is reality. In other words, those who have mastered the art of being perceived as elite reap benefits, though this has nothing to do with actual results.

At GitLab, "no ego" means that we foster and support an environment where results matter, and you're given agency to approach your work in the way that makes sense to you. Instead of judging people for not approaching work in an agreed-upon way, "no ego" encourages people to glean inspiration from watching others approach work in new and different ways.

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.

Don't let each other fail

Keep an eye out for others who may be struggling or stuck. If you see someone who needs help, reach out and assist. This might involve offering to [pair program](#) or setting up a sync brainstorming session. The goal is to connect them with someone else who can provide expertise or assistance. We are a team, so we succeed and shine together by supporting each other!

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 giving you feedback 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](#). The Bolt Handbook refers to this as the [Founder Mentality](#), where all team members should approach the problem as if they own the company.

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 feeling comfortable letting others contribute to our domain. For example, pointed, respectful feedback to a [proposal](#) by GitLab's CEO led to his own merge request being closed. However, it is not required to respond to comments.

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](#).)

Collaboration is not consensus

When collaborating, it is always important to stay above radar and work [transparently](#), but collaboration is [not consensus](#) and disagreement is part of collaboration. You don't need to ask people for their input, and they shouldn't ask you "Why didn't you ask me?". You don't have to wait for people to provide input, if you did ask them. You don't need to have everyone agreeing to the same thing - they can [disagree, commit, and advocate](#). [Two-way doors decisions](#) can be reversed as part of [disagree, commit, and advocate](#), while one-way door decisions benefit from more input. Recognize these reversible two-way door decisions for when less input is required to iterate faster. We believe in permissionless

innovation — you don't need to involve people, but everyone can contribute. This is core to how we [iterate](#), since we want smaller teams moving quickly rather than large teams achieving consensus slowly.

Collaboration Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate collaboration when we take action to help others and include other's (both internal and external) input (both help and feedback) to achieve the best possible outcome.

GitLab Job Grade	Demonstrates Collaboration Competency by...	Knowledge Assessment
5	Develops collaboration skills by learning from other team members	Knowledge Assessment for Individual Contributors
6	Grows collaboration skills by using different types of communication; files issues appropriately, asks in the right Slack channels and uses the right labels.	
7	Models collaborative behavior for fellow team members and others within the group.	
8	Coaches team members on how to collaborate more effectively and pointing team members to the right channels to collaborate.	Knowledge Assessment for People Leaders
9	Fosters collaborative decision making and problem solving across the departments.	
10	Drives team collaboration across divisions/departments, silos, and division boundaries.	
11	Develops networks and builds partnerships, engages in cross-functional activities; collaborates across boundaries, and finds common ground with a widening range of stakeholders. Utilizes contacts to build and strengthen internal support base	

- 12 Leads collaboration and teamwork in daily routines, prioritizing interactions, information sharing, and real time decision making across divisions/departments. Encourages greater cross-functional collaboration among e-team leaders.
- 14 Champions collaboration and teamwork into daily routines, prioritizing interactions, information sharing, and real time decision making across divisions/departments. Champions cross-functional collaboration among e-team leaders and GitLab.



Results for Customers

We exist to help our customers achieve more. Everything we do should be in service of making our customers successful with GitLab. Results for Customers is at the top of our values hierarchy, as our customers achieving results drives overall business performance that enables everything else.

The Results for Customers value is displayed through the following operating principles:

Set Ambitious & Measurable goals

While we iterate with small changes, we strive for large, ambitious results. We have an ambitious [mission](#) and [vision](#), and we aim to be the best in the world across all our functions. Setting ambitious, measurable goals enables us to best deliver customer results. We agree in writing on measurable goals. Within the company we use [OKRs](#) to stay accountable. We have and report against [KPIs](#) with guiding targets.

Understand our customers

All GitLab team members should understand our customers' needs, issues, and value propositions. We understand how they use GitLab and what they need from a platform in order to meet their goals. Internally facing teams consider the impact of their work as it pertains indirectly to GitLab's customers.

We better understand customers and their needs through:

- Reviewing public facing GitLab issues from our customers and users
- [Dogfooding](#) our product to understand the user experience
- Reading customer stories from Marketing and Sales
- Attending Customer fireside chats

- Learning feedback from our customers and users on product features and roadmap

Co-create

We create together with our customers. There is an open dialogue between GitLab and our customers so that we can better identify what they need. As a result of building a solution for them, we can also bring that solution to the world.

Keep end users in sight

Our focus is to increase customer results. At GitLab, one way to drive customer results is through platform enhancements that drive the most value for direct users. This requires being aware of [the Concur effect](#).

[Arvind Narayanan](#), a Princeton Professor, described his frustration with Blackboard in a viral Tweet:

It has every feature ever dreamed up. But like anything designed by a committee, the interface is incoherent and any task requires at least fifteen clicks (and that's if you even remember the correct sequence the first time).

Software companies can be breathtakingly clueless when there's a layer of indirection between them and their users. Everyone who's suffered through Blackboard will have the same reaction to this: try having less functionality!

[Ryan Falor](#) followed up on Narayanan's tweet with his definition of the Concur Effect:

1. decision makers are not direct users
2. features are overwhelming and disjointed
3. user experience gets worse over time

See [the Hacker News discussion](#) for a specific UX example.

At GitLab, we want to drive customer results through focusing on platform enhancements that drive the most value for direct users.

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-managed delivery of GitLab. This does not mean that we will agree to every feature request, but we won't let existing plans be an obstacle to working on what will drive the most customer value.
2. **Large customer requests.** Catering to requests from large customers leads to the [innovator's dilemma](#), we need to also focus on results for small and future customers.
3. **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.

4. **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 multiple customers to ensure we create scalable, highly relevant solutions.
5. **What we control.** We strive to provide the best possible experience for each of our customers, and take responsibility for all of the aspects that we can reasonably control.

Measure impact, not activity

We care about what you achieve: the code you shipped, the needle you moved, 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, unless it negatively impacted a goal or result they were responsible for. You don't have to defend how you spend your day if you are performing and delivering against expectations. We trust team members to do the right thing instead of having rigid rules. We trust team members to show up and do their best work. 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.

Dogfooding

We [use our own product](#) in the way our users do to surface improvements that will lead to better [customer results](#). GitLab is a DevSecOps Platform that can be used by people throughout the business. This is how we use it within GitLab. For example, we use our OKR functionality company-wide to inform product enhancements and for team members to have a great understanding of the customer experience. We also dogfood in the following ways:

1. Our development organization uses GitLab.com to manage the DevOps lifecycle of GitLab itself.
2. All team members use GitLab to collaborate on this handbook.
3. We capture 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.

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.

Challenger mindset

Challenging the status quo can lead to remarkable results - we must never stop. A [challenger mindset](#) requires that we continually ask ourselves bold, difficult questions about our business and the problems we solve, while resisting complacency. To succeed we must innovate and delight our customers with the value of the products we build. A challenger mindset requires a relentless pursuit of excellence - we must be [tenacious](#). Each win for our customers builds reputational capital we can use to earn the trust of prospects in a competitive market. While competition is a feature of capitalism, internally as GitLab team members, we must focus our efforts inwardly on achieving our very best results for customers to win market share.

Growth mindset

You don't always get results and this will lead to criticism from yourself and/or others. We believe our talents can be developed through hard work, targeted training, learning from others, on-the-job experience, and receiving input from others. It is in our DNA as a company and individuals to look for opportunity, stay humble, and never settle. We try to hire people based on [their trajectory, not their pedigree](#). We also strive to foster a culture of curiosity and continuous learning where team members are provided and proactively seek out opportunities to grow themselves and their careers. We believe that with the right expectations and direction, people can grow to take on new challenges and surpass expectations.

Cross-functional optimization

Our definition of cross-functional 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. For example, you may have set a non-urgent functional milestone that is supposed to land at the end of the quarter. If delivering within the last week requires engagement from [the GTM teams](#), the right decision may be to push your own team's target by a week to reduce the ask for the GTM team as the GTM focuses on meeting its revenue objectives.

In the context of [collaboration](#), if anyone is blocked by you on a question, your approval, or a merge request review, you should prioritize unblocking them, either directly or through helping them find someone else who can.

Embrace 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. We value the ability to maintain focus and motivation when work is tough and asking for help when needed.

Have Ownership & Accountability

We expect team members to complete tasks that they are assigned. You are responsible for executing with attention to detail, connecting the dots across the organization and anticipating and solving problems. As an owner, you are responsible for overcoming challenges, not suppliers or other team members. Take initiative and proactively inform stakeholders when there is something you might not be able to solve.

Sense of urgency

Time gained or lost has compounding effects. Try to get the results as fast as possible, but without compromising our other values and [ways we communicate](#), so the compounding of results can begin and we can focus on the next improvement.

Operate with a 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. Try to get results as fast as possible, but without compromising our other values and ways of working

Disagree, commit, and advocate

When a decision is in place, we expect people to commit to executing it. Any past decisions and guidelines are open to questioning as long as you act in accordance with them until they are changed. This is [a common principle](#). Every decision can be changed; our [best decision was one that changed an earlier one](#). In a manager-report relationship, usually the report is the [Directly Responsible Individual](#) (DRI). The manager may disagree with the final decision, but they still commit to the decision of the DRI.

In a group setting, participants may disagree with a proposal but not articulate their views for one reason or another. Sometimes, [many or all individuals may disagree yet choose not to speak up](#), because no one believes they would get agreement from the group. As a result, everyone loses out on their feedback. [Dissent](#) is expression of that disagreement. However, it can be difficult and even socially expensive. Expression of feedback is a way for everyone to grow and learn, and is [based on facts rather than opinions](#). Share your

perspective, rather than agreeing simply to avoid conflict or to go along with everyone else.

When you want to reopen the conversation on something, show that your argument is informed by previous conversations and [assume the decision was made with the best intent](#). 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 [DRI](#) who can change the decision instead of someone who can't.

Escalate to unblock

If there is a disagreement and you can't move forward because of it, agree to escalate and escalate to one or both of your managers. Early escalation, delivered with context of the challenge, enables managers to function as an unblocker.

Results Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate results when we do what we promised to each other, customers, users, and investors.

GitLab Job Grade	Demonstrates Results Competency by...	Knowledge Assessment
5	Develops the skills needed to commit and execute on agreed actions.	Knowledge Assessment for Individual Contributors
6	Applies commitment to results and demonstrates ability to execute on agreed actions.	
7	Models a sense of urgency and commitment to deliver results.	
8	Coaches team members to collaborate and work iteratively towards impact with the focus on the outcome and not activity worked.	Knowledge Assessment for People Leaders
9	Fosters a culture of ownership of personal performance.	
10	Drives efficient execution of results ensuring collaboration between team members.	
11	Develops quarterly OKR's ensuring the performance and results of one or more teams.	

12 Leads the achievement of results while driving the continued alignment to our values of collaboration, efficiency, diversity, iteration and transparency.

EVP/CXO Leads the achievement of results while driving the continued alignment to our values of collaboration, efficiency, diversity, iteration and transparency.

Efficiency

At GitLab, efficiency means producing [results](#) without wasting materials, time, or energy. We optimize solutions globally for the broader GitLab community over one person or a small group. Focus on efficiency should be global in nature, not just local to a given function. Global efficiency could include efficiency with customers, candidates, and contributors as well. It is easy to prioritize consistency over efficiency because consistency is often more efficient initially and makes managing processes more efficient. We should slow down when optimizing for consistency. Taking a company-wide lens when evaluating changes will help ensure that new processes will improve efficiency for GitLab as a whole and be the best decision for the company as a whole.

When we work internally with other team members, we leverage GitLab's unique working practices and operating principles to achieve top efficiency. We do not expect people outside of GitLab to conform to GitLab's ways of working, and we will make accommodations to work effectively with them. For example, we may collaborate heavily in-person and not default to async communications.

Only Healthy Constraints

Most companies regress to the mean and slow down over time. While some changes are required as a company grows and matures, not all change is inevitable or should be allowed to passively happen. As GitLab grows, we are conscious of how we operate and how it enables our ability to continue to operate with the agility of a [startup](#). We try to limit ourselves to [healthy constraints](#).

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.

Self-service and self-learning

Team members should first [search for their own answers](#) and, if an answer is not readily found or the answer is not clear, ask in public as we all should have a [low level of shame](#). [Write down any new information discovered](#) and pay it forward so that those coming after will have better efficiency built on top of practicing collaboration, inclusion, and documenting the results.

Team members have more room to grow themselves when they are able to self-service and self-learn.

Efficiency for the right group

Optimize solutions globally for the broader GitLab community. As an example, it may be best to discard a renewal process that requires thousands of customers to each spend two hours in favor of one that only takes sixty 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. In saying this, we ask team members to weigh the cost of purchases against the value that they will bring to the company.

Consider the degree to which a purchase increases your ability to better accomplish your work and achieve business [results](#) relative to cost. Lowering overhead reduces the cost to operate the business and lets us shift spend toward other priority areas.

We have [guidelines](#) around this operating principle to help team members better understand our expensing process and expectations.

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."

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

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. Team members are given the freedom to own projects and initiatives and are trusted to see them through to a successful end.

When team members are managers of one they can have an increased work/life balance, because they are more empowered to make decisions around how they allocate their time throughout each day.

Freedom and 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. You should have clear objectives and the freedom to work on them as you see fit. Freedom and responsibility are more efficient than rigidly following a process, [or creating](#)

[interdependencies](#), because they enable faster decision velocity and higher rates of [iteration](#).

When team members have freedom and responsibility over rigidity, they have more room to help others.

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. Once you have accepted the mistake, learn from it. When team members are free to accept mistakes, they can take more calculated risks.

Move fast by shipping the minimal valuable change

We value constant improvement by iterating quickly, month after month. If a task is not the [smallest viable and valuable thing](#), cut the scope.

Embrace change

Adoption of features, user requirements, and the competitive landscape change frequently and rapidly. The most successful companies adapt their roadmap and their organization quickly to keep pace. One of the things that makes this challenging is the impact on our team. People may need to change teams, subject matter, or even who manages them. This can rightly feel disruptive. If we coach ourselves to embrace the positive aspects of change, such as increased opportunity and new things to learn, we can move faster as a company and increase our odds of success. It is important to [hold management accountable for being deliberate](#).

Efficiency Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate efficiency when we work on the right things, not doing more than needed, and not duplicating work.

GitLab Job Grade	Demonstrates Efficiency Competency by...	Knowledge Assessment
5	Develops an understanding of being a manager of 1 by taking responsibility for your own tasks and delivering on commitments. Brings up ideas for process improvements to 1:1s. Learns to write everything down as it is far more	Knowledge Assessment for Individual Contributors

efficient to read a document at your convenience than to have to ask and explain.

- 6 Develops a deeper understanding of efficiency and actively identifies process inefficiencies within the team. Seeks out ways to be more effective in their role, while also starting to mentor others in ways to work efficiently.
- 7 Models a culture of efficiency within the team where people make good, timely decisions using available data and assessing multiple alternatives. Models using boring solutions for increasing the speed of innovation for our organization and product.
- 8 Takes ownership of own team process inefficiencies, implements cross team efforts in ensuring things are running smoothly. Implements a way of working in the team where team members first search for their own answers and, if an answer is not readily found or the answer is not clear, ask in public as we all should have a low level of shame.
- 9 Takes ownership of group level process inefficiencies and guides cross sub-departments in ensuring things are running smoothly. Fosters a culture in the sub-departments where you respect others' time and promote self-service and self-learning.
- 10 Drives the framework of frugality on a department level and owns departments efforts in ensuring things are running smoothly. Drives efficient resolution of highly complex or unusual business problems that impact the department / team. Holds their managers and peers accountable for upholding this value.
- 11 Develops the framework and strategy of frugality cross division resulting in efforts ensuring things are running smoothly. Develops leaders to action on division/department/team inefficiencies. Hold their management teams accountable for upholding this value.
- 12 Leads with efficiency across the company. Ensures efficient resource allocation decisions across the company. Leads across company strategy and policy improvements that move the business towards more

[Knowledge Assessment for People Leaders](#)

efficiency. They hold their senior management and the e-group accountable for upholding this value.

EVP/CXO Champions GitLab's strategy for efficiency internally and externally. Constantly looking for efficiency improvements cross company and holding other e-group members accountable for upholding efficiency too. They are comfortable leading through frugality and accepting of mistakes.

Diversity, Inclusion & Belonging

Diversity, inclusion and belonging 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 a culture that is [inclusive](#) and supports all team members equally in the process of achieving their professional goals. We work to make everyone feel welcome and to increase the participation of underrepresented groups in our community and company.

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.

Embracing uncomfortable ideas and conversations

Part of embracing diversity is a willingness to embrace often uncomfortable conversations and situations. This concept is also at the core of inclusion and helping to eliminate the problems that are faced by certain GitLab team members who may not be in the majority.

We believe that being willing to embrace discomfort is the path forward to a safe, balanced and inclusive work place for all. Challenge yourself, challenge your own pre-set notions and ideas about different cultures or things you don't understand. When we are willing to

embrace being uncomfortable, we can focus on actually fixing the issues at hand rather than simply “appearing to care”.

Understanding the impact of microaggressions

Microaggressions are much more than merely rude or insensitive comments. They can wear people down by slowly chipping away their sense of belonging/safety/inclusion over time. What is a microaggression?

“The everyday slights, indignities, put downs and insults that people of color, women, LGBT populations or those who are marginalized experiences in their day-to-day interactions with people.” - Derald W. Sue

At GitLab we believe that everyone is entitled to a safe working space where they can express who they are and participate in conversations without worry of being spoken to in a harmful way, given that we want to encourage everyone to be mindful of what is a microaggression and be mindful of their potential impact.

Seek diverse perspectives

We believe that team members seeking feedback from a diverse group of team members, inside and outside of their group or function, leads to better decisions and a greater sense of team member belonging. For more guidance on how we define Diversity, please refer to [GitLab's definition of Diversity, Inclusion & Belonging](#). Feedback from a more heterogenous group often leads to better business outcomes as we incorporate diverse perspectives and uncover unconscious bias.

An example of this operating principle in action showcases the value of actively seeking diverse perspectives. The term “Brag Document” was used to describe when individuals documented their accomplishments. Documenting accomplishments is critical to team member development. However, team members had the **psychological safety** to raise the question of whether or not the title of the document made some feel uncomfortable. In an effort to seek a **diverse perspective**, a survey was conducted in one of the [Team Member and Advocacy Resource Group \(TMRG\)](#) channels. The poll results showed that 100% of those polled preferred a different title and the title was changed.

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. Be mindful of language and use of profanity to encourage a family-friendly environment.

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 [Internship 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 team members 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. Said differently: ["culture add" > "culture fit"](#) or "hire for culture contribution" since our [mission is that 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, inclusion and belonging, 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.

There is sometimes a grey area where advocating for diversity and political activities may intersect. Team members should use discretion in grey area communications, because a culture of belonging requires us to be respectful of the broad spectrum of views within our work environment. What does this mean in practice? Please feel empowered to share information that highlights diversity, inclusion and belonging issues and how GitLab and GitLab team members can get involved. In line with our [Code of Business Conduct and Ethics](#), avoid posting articles that reference specific political figures or parties.

While it is 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), always be aware of potential sensitivities, exercise your best judgment, and make sure you stay within the boundaries of our [Code of Business Conduct and Ethics](#).

We're a global company where perspectives and local norms may differ from culture to culture. Diversity, inclusion and belonging is about broad inclusion at a worldwide level. If

there is a question or concern, please reach out to diversityinclusion@gitlab.com or [#diversity_inclusion_and_belonging](#).

Quirkiness

Unexpected and unconventional things make life more interesting. Celebrate and encourage quirky gifts, habits, behavior, and points of view. 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 [characteristics of the people who make up GitLab and how they identify](#). 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 team members. 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 [Parental Leave](#) publicly so people don't have to ask during interviews.

Inclusive language & pronouns

Use **inclusive** language. For example, prefer "Hi everybody" or "Hi people" to "Hi guys", and "they" instead of "he/she". 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**; there is no need to dwell on it, and you can work 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 topics related to gender / sexual orientation.

Learn how to pronounce other people's names

We attach part of our identity to our names, and if it is mispronounced it can feel less inclusive. If it happens repeatedly, you may be unintentionally sending a message to that person that you are not interested in learning how to pronounce their name correctly. This applies to everyone you are in contact with: team members, customers, candidates for jobs, and anyone else.

People whose name is repeatedly mispronounced might feel unimportant or self-conscious, and might not speak up about it. Other negative behaviors include giving a person a nickname without their permission, or actively avoiding using their name in sync calls.

It might be challenging to pronounce names from a different language or culture than your own, but with some effort, name pronunciation can be learned by anyone. Some ways to achieve this are:

- Ask the person for help in a private space: "I'm sorry, I don't think I am pronouncing your name correctly. Can you help me with the correct pronunciation?"
- Use the written and recorded pronunciation tools in Slack.
- Use online tools such as videos recorded on YouTube or [NameShouts](#).
- Practice the pronunciation with a friend or team member who knows the correct pronunciation.
- Always avoid making jokes or comments about how it is difficult to pronounce their name.

Use of nicknames

Some people might choose to use a nickname, for example: "Bob" instead of "Robert". As long as this is their choice this is perfectly acceptable. We should avoid assigning a nickname to a person without their permission.

Slack pronunciation features

Slack has two features to help with this issue: the phonetic name pronunciation field and the ability to record your own name pronunciation audio clip. We encourage all team members to complete both of these. Update them by [editing your profile](#).

Inclusive interviewing

This is documented on our page about [interviewing](#).

Inclusive meetings

Be consciously inclusive in [meetings](#) by giving everyone present an opportunity to talk and present their points of view. This can be especially important in a remote setting.

With internal meetings, consider using an agenda document for questions. For example, with GitLab [AMAs](#), every meeting has a numbered list that GitLab team members can add questions to. During the meeting, questions are answered in turn and discussions noted in the same document. Sometimes, these documents can have so much traffic (during the meeting) such that only a limited number of people can edit the document. In these situations, those who have questions should post on zoom chat and those who can edit the document should help copy the question over to the document. In addition, those who can edit the document should also post in zoom chat to see if anyone has any questions that they could help add to the document so that meeting attendees are more empowered to contribute to the conversation.

Customers are not used to working in this way. To promote inclusion with customers: ask participants for their goals; make sure during demos that you pause for question; leave time for discussion.

Inclusive and fair policy to regions with fewer employees

Being globally distributed has the benefit that someone can cover for you when you are off work. However, population density is not balanced across timezones. Policies should remain fair to those in less dense regions.

For example, the Asia Pacific region covers more timezones but has fewer team members. If we use an algorithm to assign tasks to those in later timezones, all American tasks would fall on the fewer Asia Pacific employees. This can damage belonging and inclusivity and should be avoided.

When planning an event, the organizer should cater for location density differences to maximize participation in all regions.

See Something, Say Something

As a globally-dispersed company, we have team members 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.

Embracing Neurodiversity

[Neurodiversity](#) refers to variations in the human brain regarding learning, attention, sociability, mood, and other mental functions. There are various neurodevelopmental conditions, like autism, ADHD, dyslexia, dyscalculia, dyspraxia, cognitive impairment, schizophrenia, bipolarity, and other styles of neurodivergent functioning. While neurodivergent individuals often bring [unique skills and abilities](#) which can be harnessed for a [competitive advantage](#) in many fields (for example, [cybersecurity](#)), neurodivergent individuals are often discriminated against. Due to non-inclusive hiring practices, they sometimes have trouble making it through traditional hiring processes. Neurodiversity inclusion best practices benefit everyone, and at GitLab, everyone can contribute. The handbook, values, strategy, and interviewing processes must support the ability for everyone to thrive.

At GitLab we embrace Neurodiversity through adopting a variety of different work styles and communication styles, and we lean into [transparency](#), asynchronous as a default working style, and pre-filled meeting agendas. These best practices become even more important when embracing neurodiversity. Providing multiple ways to consume information (written / video / audio) allows everyone to contribute independent of their preferred comprehension style. It is important to ask team members specifically what their preferred communication method is in order to provide them information in a format that is easily consumable for them.

Remember, **brains work differently** and always [assume positive intent](#), even if someone behaves in an unexpected way. While it may be an unexpected behavior to you, it may not be unexpected to the individual exhibiting the behavior. That is the beauty and value of diversity, embracing differences and becoming stronger and better as a result.

We also recommend that all team members review the [Reasonable Accommodation](#) process. A Reasonable Accommodation for a team member could include noise-cancelling headphones, scheduling smaller group session zoom calls, providing very explicit and precise instructions and due-dates when given tasks, or providing a variety of supportive software tools.

The most important thing that managers can do is facilitate an environment in which all team members feel [psychologically safe](#) enough to make requests for [what they need](#) in order to do their job.

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 five days after a hurricane: "THANK YOU to GitLab for providing a culture where "family first" is truly meant". Use the hashtag: **#FamilyAndFriends1st**

Equity not just equality

[Equity vs. Equality: What's the Difference?](#)

While the terms equity and equality may sound similar, the implementation of one versus the other can lead to dramatically different outcomes for marginalized people.

Equality means each individual or group of people is given the same resources or opportunities. Equity recognizes that each person has different circumstances and allocates the exact resources and opportunities needed to reach an equal outcome.

Diversity, Inclusion & Belonging Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate diversity, inclusion and belongings when we foster an environment where everyone can thrive and ensuring that GitLab is a place where people from every background and circumstance feel like they belong and can contribute.

If you would like to improve your skills or expand your knowledge on topics relating to Diversity, Inclusion, & Belonging at GitLab, check out our resources:

- [Being an Ally](#)
- [Being Inclusive](#)
- [Recognizing Bias](#)

GitLab Job Grade	Demonstrates Diversity & Inclusion Competency by...	Demonstrates DIB Behaviors by... (Should not be considered an exhaustive list)	Knowledge Assessment
4	Learns to understand the impact of biases. Gathering more information about the skills needed to be accountable for their actions, apologizes and learn.		
5	Develops an understanding of the impact of biases; seeks to learn more about their own biases. Is accountable for their actions, apologizes and learns from their mistakes.	DIB training and/or other company wide training to further education on DIB <ul style="list-style-type: none"> • Attend DIB Initiatives Calls to stay informed and	Knowledge Assessment for Individual Contributors

6	Has a growing understanding of the impact of biases; fosters a sense of inclusion and belonging on their team. Holds themselves and peers accountable for upholding this value by kindly pointing out when mistakes might be made. Encourages an inclusive team environment where differences are encouraged and everyone can contribute.	connected with ongoing efforts and discussions	
7	Actively aware of how bias or exclusion might occur on a team and helps to facilitate a team environment where team members belong and feel safe. Models empathy with their interactions with customers and cross functional team members.	<ul style="list-style-type: none"> • Participate in a DIB Initiative: join TMRGs you identify with, support other groups as an Ally, attend events, "like" slack posts and spread the word • Participate in a TMRG initiative • Member of a working group related to a DIB initiative, e.g. participate in a Mentorship program 	
8	Implements best practices to limit bias on their team. They ensure blameless accountability is practiced throughout their team. Creates an environment where team members feel safe to share ideas and welcomes individual differences.	<p>DIB training and/or other company wide training to further education on DIB</p> <ul style="list-style-type: none"> • Attend DIB Initiatives Calls to stay informed and connected with ongoing efforts and discussions • Participate in a DIB Initiative: join TMRGs you identify with, support other groups as an Ally, attend events, like posts and spread the word • Participate in a TMRG initiative 	Knowledge Assessment for People Leaders
9	Proactively finds ways of facilitating an inclusive team environment and assesses processes to protect against unconscious bias. They hold		

	<p>their team members accountable including cross functional stakeholders. Promotes individual differences across their team and other departments.</p>	<ul style="list-style-type: none"> • Member of a working group related to a DIB initiative, e.g. participate in a Mentorship program • Hiring Manager ensure a diverse candidate slate and interview panel • Active participant and advocate for department DIB goals
10	<p>Drives diversity, inclusion and sense of belonging across their department. They hold their managers and peers accountable for upholding this value. They are actively involved in the execution of D&I strategies and encourage others to participate.</p>	<p>Black is Tech, Grace Hopper</p>
11	<p>Embeds the value of Diversity & Inclusion across their division and finds opportunities to limit the impact of bias on decision making processes. Uses feedback and data to formulate a strategy on how to make improvements. They hold their management teams accountable for upholding the value.</p>	
12	<p>Leads with the value of Diversity & Inclusion across the company and finds opportunities to limit the impact of bias on decision making processes. They sponsor internal initiatives to increase</p>	<p>DIB Team & Leadership DIB Council to establish an action plan for your departments & division</p>

	trust, psychological safety and inclusion. They hold their senior management and the e group accountable for upholding this value.	<ul style="list-style-type: none"> • Serve as an TMRG executive sponsor • Embed DIB into All hands or in person events e.g.review organizations OKR as it relates to DIB progress, champion trainings related to further DIB knowledge, invite guest speakers to advocate
EVP/CXO	Champions the value of Diversity, Inclusion and Belonging into the company's strategy. They champion and sponsor internal and external D&I initiatives. They speak to the importance of this value in company-wide meetings. They hold their leaders and other e group members accountable for upholding this value. They continuously seek ways to increase trust, psychological safety and inclusion across the broader company.	<p>DIB Team & Leadership DIB Council to establish an action plan for your departments & division</p> <ul style="list-style-type: none"> • Serve as a TMRG executive sponsor • Embed DIB into All hands or in person events e.g.review organizations OKR as it relates to DIB progress, champion trainings related to further DIB knowledge, invite guest speakers to advocate

Iteration

[Merriam-Webster](#) defines iteration as the “the action or a process of iterating or repeating: such as a procedure in which repetition of a sequence of operations yields results successively closer to a desired result.” At GitLab, we iterate to do the [smallest valuable thing to get fast feedback and efficiently reach a desired end goal](#). Feedback can be from

internal users (dogfooding), a limited number of external users (through our [early access program](#)), or through feedback from our broader user community. We validate each iteration and adjust, but not at the expense of the user experience that we deliver to our customers.

When we iterate at GitLab, we break up the work that we know we need to do into smaller chunks to iterate toward a targeted end state:

1. Merge in codebase
2. [Dogfood](#)
3. Have some external users (early access program)
4. Ensure global optimization (use standardized systems)
5. Plan beyond the iteration

Iteration does not require us to ship features that are open to all users from day one. Feedback can come from internal users or a limited number of external users (early access program). Moving through the [release process is not iteration](#) though. Iteration is also not a [replacement for having a plan](#). We expect you to know where you are going, but you can iterate to get there.

An iteration might be additive (adding something) or subtractive (removing something). If you make suggestions that can be excluded from the first iteration, turn them into a separate issue that you link.

While you should have a clear vision of the desired outcome and how it addresses a customer pain point or improves the user experience, be efficient in your planning. Unless you identify important cross-functional interdependencies, focus detailed planning on the first step. It might feel you are moving too slowly; however, planning is critical in order to ensure you can move fast when implementing. You're doing it right if you feel that you have shipped the minimal feature set 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. Frequently, the simplest version that provides value turns out to be the best one.

Many people who join GitLab say they already practice iteration. But this is the value that is the hardest to understand and adopt. People are trained that if you don't deliver a perfect or polished thing, there will be a problem. 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 GitLab team members 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. In reality, it is possible to ship a minimally valuable product while continuing to adhere to the documented quality standards.

The way to resolve this is to write down only what value you can add 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. Reverting work back to a previous state is positive, not negative. We're quickly getting feedback and learning from it. Making a small change prevented a bigger revert and made it easier to revert.

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".

Iteration enables [results](#) and [efficiency](#).

How to do iteration with Carol Teskey & Sid



In the [GitLab Unfiltered video](#) embedded above, GitLab co-founder Sid Sijbrandij shares key operating principles to reinforce iteration in an organization.

Start with a long-term vision

Iteration involves driving results in pursuit of a long-term vision. While the intermediate goals may change as we iterate, we are unlikely to be successful if we don't start with a vision of what we are working toward. Shipping that vision in iterations allows us to learn from customers using it and adjust the vision if needed. Iteration for the sake of iteration can lead to inefficiencies and not deliver desired results.

Iteration is no substitute for planning

Iteration without a plan can lead to inefficiencies and a subpar customer experience. Before iterating we need to plan. A plan should include:

1. Time-bound objective: Where we want to be in a year
2. UX: User experience we are working toward
3. Quality: What quality is sufficient quality, inclusive of security
4. Success metric: Usage we want at a specific time
5. Data schema: The data schema we need to measure progress towards project goals
6. GTM plan: How we want to go to market
7. Enablement: When we'd plan to train and enable the support and field teams
8. Marketing: When we'd launch marketing (doesn't have to be at release)
9. Secure by design: default to the most secure configuration

The release process is not iteration

Moving through the release process is not iteration.

The release process can include:

- [Dogfooding](#)
- [Early access](#)
- Incremental release using [a feature flag](#)
- [Development stage progression](#) (such as experiment to beta)
- Release
- Announcement

While [development stages](#) can be used to indicate release progress, is not itself iteration.

Iterate toward global maximum

If we are not aware of interdependencies beyond our team, and we are not collaborating with others across the organization, we risk deliverables that settle into a “local maximum” of quality, richness, and efficiency. This localization is largely defined by team structure and organizational boundaries. While an iteration can take place within a single team, that team is responsible for identifying inter-dependencies and proactively communicating and aligning with other teams working on related projects. This helps ensure that iterations are not “half-baked” and align with work being done across the entire organization.

Don't wait

Don't wait on the small things. 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 [GitLab Summit](#)). Inventory that isn't released is a liability since it has to be managed, becomes outdated, and you miss out on the feedback you would have received had you implemented it straight away. When we don't wait we signal intent to others that we have a purpose to resolve something. **Note:** "Don't wait" should not be used as a justification for not iterating toward the global maximum or at expense of the plan. If there are interdependencies to be considered or the iteration is customer facing, slow down and ensure that we are considering what is best for GitLab and our customers.

Set a due date

We always try to set a due date. If needed, we cut scope. If we have something planned for a specific date, we make that date. For example we [shipped over 133 monthly releases](#). But every one of them doesn't contain all the features we planned. If we planned an announcement for a certain date, we might announce less or indicate what is still uncertain. But we set a due date because having something out there builds trust and gives us better feedback.

Cleanup over sign-off

As discussed in [Sid's interview on iteration](#), waiting for approval can slow things down. We can prevent this with automation (such as tests of database migration performance) or clean-up after the fact (refactor a Pajamas if something was added that isn't coherent), but we try to ensure that people don't need to wait for sign-off. As iteration does not require us to ship to all users on day one, we can clean up after an internal or beta release to mitigate the negative impact to all customers.

Start off by impacting the fewest users possible

Iteration does not mean being open to all users from day one. If you do a gradual rollout of your change, prefer:

- Few users over many users
- Internal users (dogfooding) over external users
- Environments with fast feedback (SaaS) over slow feedback (self-managed)

Reduce cycle time

Short iterations reduce [our cycle time](#). Merging frequently also prevents merge conflicts.

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.

Minimal Valuable 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. This may be additive (adding something) or subtractive (removing something). 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, but please state that you couldn't think of a good solution and list any solutions you considered.

By making a proposal you also provide better visibility into the work and the context surrounding it.

In this [GitLab Unfiltered video](#), GitLab co-founder Sid Sijbrandij converses about iteration in engineering, leveraging proposals to break work into smaller components.

Everything is in draft

At GitLab, we rarely mark any content or proposals as drafts. Everything is always in draft and subject to change. When everything is in draft, contributions from team members as well as the wider community are welcomed. By having everything in draft and [assuming others have low context](#), confusion can be reduced as people have shared access to information.

Under construction

As we continue to expand the number of users we have, they will continue to expect stability and reliability. We must optimize for the long term without sacrificing stability along the way. This means that users may be inconvenienced in the short term, but current and future users will enjoy a better product in the end.

Educating users on the longer-term plan helps create a shared understanding of how a small change will incrementally grow into something more. For example, we could share how a dropdown will evolve into a much more nuanced solution in the future. We can take the following steps to articulate our plan:

1. Open a feedback issue that provides context about the initial MVC ([example](#))
2. Ensure the direction page articulates a long-term plan ([example](#))
3. Announce the MVC in a release post, link to the feedback issue, and link to the direction page ([example](#))

Low level of shame when dogfooding

In many organizations, you take a risk when you put forth any work that's not perfect, work where you haven't spent endless cycles planning for contingencies or counterpoints. Because of this, you're incentivized to invest a lot of time and effort into preparing for 'What if?' scenarios before any work is presented, even if the release is not customer facing and there is a low level of risk in imperfection.

The downside to that is clear when we are dogfooding: If you do eventually put forth the work, but it needed to be course-corrected a long time ago, you've squandered time that you could have spent improving it through iteration.

Having a low level of shame when dogfooding or working internally requires you to combat a natural inclination to conceal work until it's perfect, and instead celebrate the small changes.

Cultural lens

Cultural differences can bring unique challenges and expectations to iteration. For some, expressions like "it doesn't have to be perfect..." can challenge cultural norms. We encourage you to bring your authentic self and seek shared understanding when iterating. [Giving feedback](#) and ensuring [psychological safety](#) are necessary for every iterative attempt.

Focus on improvement

We believe great companies sound negative because they focus on what they can improve, not only on what is working well. In every conversation, inside and outside the company, we should ask a question: 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 **Short Term Critical And Long Term Optimistic** (STeCALT0, for short).

Be deliberate about scale

First, optimize for speed and results (and be deliberate about how your change affects other processes/functionality); when it is a success, figure out how to scale it. Great examples are in [this article by Paul Graham](#).

Resist bundling

Resist the urge to bundle a series of smaller iterations so team members don't see a project as their last (or best) opportunity to contribute. It's tempting to [create encompassing projects or initiatives](#) that roll many smaller projects up. This incarnation of scope creep drives up cost, encourages fewer risks, and incentivizes perfection (via longer cycle times) over progress. When we resist bundling, we reduce the risk that work will be canceled due to scale or scope. By resisting bundling we also reduce the coordination needed because fewer people or teams may be involved.

Make two-way door decisions

Most decisions are easy to reverse. In these cases, the [Directly Responsible Individual](#) should go ahead and make them without approval. Only when you can't reverse them should there be a more thorough discussion. By [embracing iteration](#) and making two-way door decisions, we are more efficient and achieve more results.

Changing proposals isn't iteration

Changing something without shipping it is a revision, not iteration. Only when the change is rolled out to users, whether [internal users or a limited customer group](#), 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. Never call a revision an iteration because it is almost the opposite.

Embracing Iteration

In order to embrace iteration, we should have the attitude that we are trying to achieve as much as possible in a small amount of time; it's where we land at the end state of an iteration that counts. The benefit of iteration is to get fast feedback from users. Focus on sharing context at the **end of the first iteration** rather than a **hypothetical future state** that requires multiple iterations. By embracing iteration we can increase creativity in incremental components.

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 via [Handbook Usage guidelines](#), and then add additional sections iteratively with subsequent merge requests. Similarly, when adding features to GitLab, consider ways to [reduce the scope](#) of the feature before creating the merge request to ensure your merge request is as small as possible.

Always iterate deliberately

Rapid iteration can get in the way of [results](#) if it's not thought out; for example, when adjusting our marketing messaging (where consistency is key), product categories (where we've set development plans), [organizational structure](#) or [product scope alignment](#) (where real human stresses and team stability are involved), 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 in the [GitLab Handbook Usage](#) page and takes place via merge request approvals.

12 things that are not iteration

Iteration is often counterintuitive and difficult to do. To clarify what an iteration is, it helps to see examples of what is not an iteration. Below are 12 examples of things we've seen mistaken as iteration, but don't meet our definition of iteration.

1. Reducing quality, or lowering goal posts
2. Avoiding or reducing documentation
3. Compromising on security
4. Delivering something that's not the recommended path or on by default
5. Shipping something of no value
6. An excuse to focus on unimportant items
7. [Moving through the release process](#)
8. Revisions you don't ship or publish
9. An excuse to impose unrealistically tight timelines
10. [An excuse to avoid planning](#)
11. Imposing long hours
12. Expecting others to fix your work

In this [GitLab Unfiltered video](#), GitLab co-founder Sid Sijbrandij elaborates on each of these 12 things that are not iteration.

Iteration Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate iteration when we do the smallest viable and valuable thing, get it out quickly for feedback, and make changes based that feedback.

GitLab Job Grade	Demonstrates Iteration Competency by...	Knowledge Assessment
5	Develops own knowledge by trying and failing. When asking questions isn't content with silence or unhelpful/incomplete responses, seeks out primary sources.	Knowledge Assessment for Individual Contributors
6	Actively looks for opportunities to iterate and contribute to boring solutions. Balances short term gains and long term benefit with team's help. Ships things that aren't 100% knowing that you'll be able to improve them in the next revision. Asks questions with abandon. Publicly shares failures if you'll help colleagues learn.	
7	Independently balances short term gains and long term benefit. Identifies opportunities to deliver projects in an iterative way.	
8	Is able to take long term goals and turn them into small actionable steps that can be implemented in an iterative way. Identifies and prevents decisions that are not "two-way door decisions". Ships. All the time. Sounds like a broken record in discussions with more junior members of the team; always asking if we can make something smaller.	Knowledge Assessment for People Leaders
9	In addition to upholding the requirements of a Staff/Manager level, a Principal/Sr. Manager practices and fosters the value of iteration to team members. They hold their team members accountable for iteration and boring solutions.	
10	In addition to upholding the requirements of a Principal/Sr. Manager, a Distinguished/Director proactively finds ways to drive the value of iteration and boring solutions.	
11	In addition to upholding the requirements of a Distinguished/Director , a Sr. Distinguished/Sr. Director	

embeds the value of Iteration across the department and division. They use their cognitive and analytical abilities to anticipate and adapt to unpredictabilities in regard to strategic risk in a way that benefits all involved.

12 In addition to upholding the requirements of a Sr. Distinguished/Sr. Director , a Fellow/VP leads the way for the value of Iteration across the division and cross functional teams. They confidently lead their teams through change and proactively take risks based on values and the strategic vision.

EVP/CXO In addition to upholding the requirements of a Fellow/VP, the EVP champions the value of Iteration across GitLab. They are comfortable leading through discomfort and the unease associated with change and innovation.

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. Transparency is not communication. Just because something exists in the handbook or elsewhere doesn't mean it can't be communicated again or in a more robust fashion to the people who need to understand or acknowledge it. On a personal level, be direct when sharing information, and admit when you've made a mistake or were wrong. When something goes wrong, it is a great opportunity to say "What's the [kaizen](#) moment here?" and find a better way without hurt feelings.

Even as a [public company](#), we know that our value of transparency will be key to our success. This value can be hard to follow at times. You might ask yourself: what should be shared, how much to share, whether or not to speak up but definitely take the time to always opt for maximum transparency by adhering to the operating principles below. 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. As a [public company](#), we declare everyone in the company as an insider, which allows us to remain transparent internally about our numbers, etc. Everything else that can be transparent will continue to be so.

When there are exceptions, material that is [not public by default is documented](#).

Public by default

Everything at GitLab is public by default. 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.

One example of transparency at GitLab is the [public repository of this website](#) that also contains this [company handbook](#). Others include the [GitLab CE](#) and [GitLab EE](#) issue trackers, as well as [marketing](#) and [infrastructure](#). Transparency creates awareness for GitLab, allows us to recruit people that care about our values, 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.

In line with our value of transparency and being public by default, all GitLab team member [profiles](#) should be public. Public profiles also enable broader collaboration and efficiencies between teams. To do so, please make sure that the checkbox under the [Private profile](#) option is unchecked [in your profile settings](#). If you do not feel comfortable with your full name or location on your profile, please change it to what feels appropriate to you as these are displayed even on private profiles.

Because we are public by default and have the [SAFE framework](#) we don't need to make cases for why things should be transparent. If something is unSAFE and needs to remain [not public](#) it can be.

Not public

We make information public by default because [transparency is one of our values](#). However it is [most important to focus on results](#). Therefore, a category of information is **public** unless there is a reason for it not to be. 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 GitLab Legal and Corporate Affairs believes it carries undue risk. We document what is [not public by default](#) on our communication page.

If you believe something shouldn't be public that currently is (or vice versa), then [make a merge request](#) to the relevant page(s) suggesting the change so that you can collaborate with others and discuss with the [DRI](#). When content contains information which is [not public](#) it is recommended to remove the specific sections which are not public, put them on their own page in the internal handbook, and then link out to that with a "not public/internal only" note. Always share publicly what we can.

When information is not public, it may also be treated as limited access, only shared with certain GitLab roles, teams, or team members due to privacy considerations, contractual obligation, or other reasons that the author or DRI can specify. Certain kinds of information default to limited access, including details about team members or customers who did not give permission to share the information.

Most companies become non-transparent over time because they don't accept any mistakes. Instead, we should always err on the side of transparency when there is a choice to be made between caution or inaction, and transparency. If we make a mistake, we now know what the limits of transparency are for the company and we should [document this](#). *The only exception to this rule would be in the case when there are legal concerns.*

Because some information is [not public](#) the public information can be lacking some context. We should be cognizant of that.

Directness

Being direct is about being transparent with each other. We try to channel our inner [Ben Horowitz](#) by being [both straightforward and kind](#). Feedback is always about your work and not your person. That doesn't mean it will be easy to give or receive it.

Articulate when you change your mind

If you state one thing, and then change course and support a different direction, point, or outcome, articulate this. It is OK to have your position changed by new data. Articulating that an *earlier* stance is not your *current* stance provides clarity to others and encourages data-driven decision making.

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.

Transparency is most valuable if you continue to do it when there are costs

We practice transparency even when hiding the facts would be easier. For example, many companies 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.

Transparency has costs (distraction, mis-interpretation, etc.) but also great benefits (productivity, hiring, retention, brand awareness, etc). We should carefully weigh the tradeoff between costs and benefits, to prevent a knee-jerk reaction to reduce transparency when it has costs.

Single Source of Truth

By having most company communications and work artifacts be public to the Internet, 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.

Findability

Our transparency value means more than just making information accessible to all. In order to [improve performance](#) it's important that we not only ensure information is accessible, but also ensure it flows to the correct places and is [findable](#) by those who need it.

Focusing on information flow will ensure you, for example, utilize [multi-modal communication](#), or that you keep your [stakeholders informed of changes](#) by posting links to MRs in Slack.

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.

This also helps with institutional memory: a year from now when you want to know why a decision was made, or not, the issue or MR that has the decision also shares why the decision was made. This is related to [Chesterton's fence](#) - it's much easier to suggest removing or changing something if you know why it exists in the first place.

If you use generalized terms such as "industry standard" or "best practices," be sure to give context, as without context they can be seen as potentially vague or opaque.

Similarly, merely stating a single value isn't a great explanation for why we are making a particular decision. Many things could be considered "iteration" or "efficiency" that don't match our definition of those values. Try to link to an operating principle of the value or provide more context, instead of just saying a single value's name.

Saying why and not just what enables discussion around topics that may impact more than one value; for instance, when weighing the [efficiency of boring solutions](#) with the focus on [customer results](#). When decisions align with all of our values, they are easy to discuss and decide. When there are multiple values involved, using our [values hierarchy](#) and [directly](#) discussing the tradeoffs is easier with more context.

Articulating why also helps people understand how something changed when you [articulate that you changed your mind](#).

Saying why does not mean justifying a decision against all other suggestions. The [DRI](#) is responsible for their decision. The DRI is not responsible for convincing other people, but they should be able to articulate their reasoning for the change.

When a GitLab Team Member comes across an ask or material (MR, handbook, etc.) that does not provide a “why” with sufficient context, the Team Member is responsible for getting the why and, if needed, working with the DRI to ensure that it is adequately documented and communicated to give context to other team members. In the absence of a why, team members may speculate the why. This is something that can lead to disruption and inefficiency.

Reproducibility

Enable everybody involved to come to the same conclusion as you. This not only involves [reasoning](#), but also providing, for example: 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](#).

Transparency Competency

[Competencies](#) are the Single Source of Truth (SSoT) framework for things we need team members to learn. We demonstrate transparency when we are open with as many things as possible reducing the threshold to contribution and make collaboration easier.

GitLab Job Grade	Demonstrates Transparency Competency by...	Knowledge Assessment
5	Uses public issue trackers, projects, and repositories when possible. Looks for opportunities to publicly share the things that they are working on.	Knowledge Assessment for Individual Contributors
6	Provides context and background on projects and issues so that those with no prior knowledge are able to contribute to the discussion. They welcome feedback and new ideas as they know that will lead to a better solution.	

- 7 Continually surfaces improvements across their functional area of expertise. They share feedback with others and understand how to disagree and commit to solutions. They model what it means to be as open as possible. They encourage conversation in public channels.
- 8 Implements open processes across their team. They also track team issues and projects openly so their team members are aware of everything that is happening on a team at a given time. They leverage feedback to drive the best possible outcomes with the information they have available. They also share feedback with their team and their peers in a timely, kind manner so their position on a given topic is known.
[Knowledge Assessment for People Leaders](#)
- 9 Fosters and coaches openness across cross functional departments. They lead cross functional issues, projects and ideas inviting feedback to generate the best possible solution. They hold their teams accountable to continue to find opportunities to share things openly. They give feedback to their team members, peers and managers in a timely, kind manner so their position on a topic is known.
- 10 Drives their departmental strategy with openness as a key value. They hold their management team accountable to working openly and pushes them to make everything transparent even when it might be difficult to do so. They coach managers on the value that additional feedback can bring to the end solution.
- 11 Develops leaders that work openly and continue to provide timely, kind feedback across their division. They develop leaders that drive their teams with openness as a foundational part of the way that they operate.
- 12 Leads the company by being open in all things. They are open with things that might traditionally not be shared broadly. They communicate directly and provide feedback in a timely manner to initiatives happening within their department and across the company. They hold the e group and other leaders accountable for upholding this value.

EVP/CXO Champions transparency both internally, across the company and externally. They participate both internally and externally in events and share the value that being open can provide to increasing trust with team members and others that interact with our product. They provide timely, kind feedback with initiatives happening internally and externally. They hold the e group and other leaders accountable for upholding this value.

Why have values

Our values provide guidelines on how to behave and are written to be actionable. They help us describe the type of behavior that we expect from GitLab team members. They help us to know how to behave in the organization and what to expect from others.

Values provide a framework for distributed decision making, detailed in GitLab's [TeamOps](#) management philosophy. They allow individuals to determine what to do without asking their manager and they allow teams to make consistent decisions. When teams across the organization reference the same values in their decision making, there is consistency in how decisions are made. This ensures that [our culture](#) remains driven by our values.

Lastly, values create a [conscious culture](#) that is designed to help you prosper and experience exceptional personal growth through work.

Five dysfunctions

Our values also help us to prevent the [five dysfunctions](#):

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

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.

Operating principles

Operating principles are behaviors that empower GitLab team members to definitively live out a given value. They clarify what a given core value means and looks like *at GitLab*. Understanding this distinction is critical to thriving at GitLab, particularly for [newer team members](#) who may be familiar with a prior organization's interpretation of iteration or collaboration (as examples).

Process for removing operating principles

Values are not just things we do, but things that actively drive good behavior. When we remove them it doesn't mean we stopped believing in it, just that it wasn't actively helping to drive behavior. If we don't prune our operating principles, then we will be like every other company: things that make sense but are not leading to a better culture.

1. To **remove** an operating principle from the Handbook page, submit your change through a merge request and explain your reasons in the merge request description.
2. The GitLab Value Handbook Page owner must approve and merge the request.

Mention the specific value

Most companies have a list of values. In companies without strong values, folks often use generalizations when they refer to values. For example, "not a value add" or "scored well on values during our interview." In companies with strong values, folks name the specific, relevant value as it applies to a given topic or situation. Values are only powerful when they are individually understood and applied by team members.

How to scale the business while preserving GitLab values?

For certain business decisions or projects (such as [compensation](#) and [end-point management](#)), GitLab team members may have a lot of opinions and interest, and they want to provide their feedback and comments. On the other hand, it might be challenging for the project DRI to digest and respond to all these inputs. What should you do in this scenario?

Everyone can contribute at GitLab. We encourage team members to share feedback and leave comments on issues. Leaving feedback and comments shows that team members care about a topic and about GitLab as a company. These perspectives may also uncover potential risks and problems in the project.

There shouldn't be a ["Don't they have their job to do?"](#) type of response. Furthermore, we shouldn't judge team members who are perceived as being the "squeaky wheel." At GitLab, we [measure impact, not activity](#). As long as a team member is producing required results, they are empowered to decide how to spend their time.

On the other hand, as GitLab grows in size, we need to make decisions and the decisions may not be agreed to by everyone. If a decision or project is sensitive or controversial, and receives large amounts of feedback, it can be challenging for the project DRI to handle. In these cases, it's best to have time-boxed feedback built into timelines.

In a hypothetical example where a DRI needs to decide between red and gold potatoes for a stew, they would create an issue with the following sentiment:

We're deciding between red potatoes and gold potatoes to go into the stew. We have to decide by Tuesday 2020-07-14 so that we can get our order to the grocery store on Wednesday 2020-07-15. We'll be collecting input and feedback until that point. Jane is the DRI and will make the decision on 2020-07-14 with all the information we have at that point. Here is the framework we're using for the decision:

- are there allergies to consider?
- cost per pound
- team member preferences

Once the decision is made, it will be what is going into the stew.

This method [has shown itself to be effective](#) at soliciting productive feedback that doesn't derail a timeline while ensuring team members feel heard.

Why our values are public

Companies are encouraged to copy and implement GitLab's values. They are Creative Commons and can be copied verbatim.

We make our values public for the same [reasons](#) we make our [OKRs](#) (Objectives and Key Results) and [strategy](#) public. There is great power and efficiency in teams who share company values. Concealing values until *after* someone is hired into an organization is not a wise strategy.

Not everyone will see our values and feel aligned with them, and that's OK. By making values public, it shows respect for the time of job seekers who conduct due diligence on

prospective employers. When people who *are* aligned with GitLab's values apply for an [open vacancy](#), this allows our hiring teams to more efficiently move candidates through the [interview process](#).

In a [GitLab Unfiltered interview on values](#), GitLab co-founder Sid Sijbrandij offers the following context.

Companies may ask you to write a blank check. They'll say, 'Come join our organization, and when you're here, you need to subscribe to our values, our way of working, and our strategy. It's very essential, and it's part of our identity!'

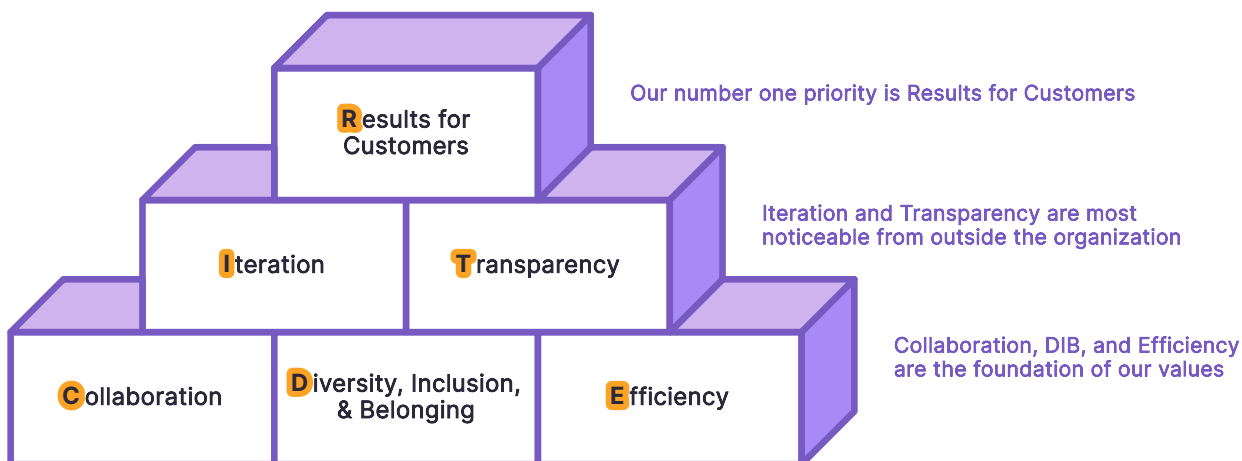
But these companies don't give you the opportunity up front to evaluate it. It doesn't make any sense to me. If it's so important that people share your values, have them out there.

Hierarchy

Occasionally, values can contradict each other. 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.

Think of the hierarchy as a weighting system. Values higher in the hierarchy do not automatically override values lower in the hierarchy. Here are some examples:

- If a change impacts Transparency positively but impacts Efficiency negatively in roughly the same amount, we would move ahead since Transparency is higher in the hierarchy than Efficiency.
- If a change has a massive positive impact on Diversity but negatively impacts Iteration, we would move ahead even though Diversity is lower in the hierarchy than Iteration because the overall impact is more positive than negative.



In a [GitLab Unfiltered interview on values](#), GitLab co-founder Sid Sijbrandij offers the following context.

It's an attempt to relieve at least some of the tension. It's not absolute. If you think of values as binary, that's not going to work. There will always be interpretation, and there's always magnitude to consider.

We made a hierarchy so that it's clear, in the end, the result matters most. For instance, we're not going to be transparent for the sake of being transparent. We're not radical in our transparency. We do it because we think it will lead to better outcomes.

Those hierarchies are really important. They won't preempt every debate, but it helps.

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 Slack channel](#). If you're not part of those groups, please send a direct X/Twitter message to [@sytses](#).

How do we reinforce our values

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

1. Criteria we use for [promotions](#) and communicate to the whole company on announcement.
2. What we select for during [hiring](#).
3. What we emphasize during [on-boarding](#).
4. Criteria we use for our [annual compensation review](#).
5. What we refer to when [making decisions](#).
6. The example the E-group sets for the company since [a fish rots from the head down](#).
7. What we expect from all team members, as [ambassadors for our values](#).
8. Keeping them [up to date](#) with a [stream of commits that add details](#).
9. Behavior we give each other [360 feedback](#) on.
10. Behavior we [compliment](#).
11. Criteria we use for [discretionary bonuses](#).
12. What we include in our [offer letters](#)

13. Criteria we use to [manage underperformance](#).
14. What we do when we [let people go](#).
15. Giving value awards during [GitLab Summit](#).
16. Providing GitLab team members and [qualified individuals](#) transparency into all aspects of the company through the [CEO Shadow Program](#) to enable them to better engage and collaborate cross-functionally.
17. Linking the takeaways of courses to our values, like we did for [the Crucial Conversations training](#).
18. The default settings of the software we use (for example: [Speedy meetings](#), [document sharing](#), agendas, etc.)
19. Reinforcing our values with features in GitLab, for example the [Iterations feature](#).
20. Applying one of our [values virtual backgrounds](#) in video calls.
21. Our GitLab [Song Book](#), the song lyrics often mention GitLab values.
22. Regularly conduct a values exercise at the [e-group offsite](#).

The most important moments to reinforce our values are decisions which affect individual team members most: hiring, promotions, and bonuses, which is why every promotion document at GitLab is shared with the entire company and uses the values as its core structure.

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

Your values are what you hire for, what you praise people for, and what you promote them for. By definition, what you do in those instances *are* your values. It's not what you say they are. Values should be explicitly part of our [hiring](#) process, our job profiles, and our [review process](#).

When we give [bonuses](#) and [promotions](#), they are always linked to values. That's the crucial thing. If you reinforce them there, that's the most powerful thing you can do. —
Sid Sijbrandij, GitLab co-founder

What to do if values aren't being lived out

Value erosion can occur when indifference and apathy are tolerated. It can also occur when individuals justify undesired behaviors by interpreting values as "me values" rather than "company values." For example, a team member may speak to the importance of personal

efficiency in order to justify not collaborating professionally with peers. This is not what we expect from team members in terms of efficiency and collaboration.

If you feel that values are not being lived out in a given scenario, speak up and ask for context in a respectful manner. Navigating value conflicts starts with [assuming positive intent](#) from other team members. Offer links to relevant values and/or operating principles when discussing the issue. If there is confusion or disagreement about the interpretation of a value, please surface the discussion in GitLab's [#values Slack channel](#) (for GitLab team members) or @-mentioning [@gitlab](#) on X/Twitter (for those who do not work at GitLab).

In a [GitLab Unfiltered interview on values](#), GitLab co-founder Sid Sijbrandij offers the following context.

Almost every time we face a hard decision at GitLab, it's because values are in conflict. It's not binary logic. It requires conversation, and sometimes there is no obvious answer. We can only achieve resolution by respectfully talking with each other and trusting the DRI to make the ultimate decision.

Permission to play

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

1. Be truthful and honest.
2. Be dependable and reliable.
3. Try to keep promises. If you might not keep a promise, proactively communicate as soon as you suspect it.
4. Be deserving of the trust of our team members, users and customers.
5. Be committed to the success of the whole organization.
6. Act in the best interest of the company, our team members, our customers, users, and investors.
7. Make the best decisions for GitLab.
8. Act in accordance with the law.
9. 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.

Playing politics is counter to GitLab values

We don't want people to play politics at GitLab.

An example of politics is people discussing a proposal and being overly focused on whose proposal it is. This is a manifestation of the [Belief Bias](#), where we judge an argument's strength not by how strongly it supports the conclusion but by how strongly we support the conclusion. Proposals should be weighed on their merits and not on who proposed them. Another example is people being promoted based on others liking them or having a lot of alliances. We want people to be promoted based on their results. We value collaboration, but that's different from being promoted just because people like you.

Below are some attributes of political and non-political work environments. GitLab plans to maintain a non-political one.

Political environment	Non-political environment
Values are weaponized and used out of their intended context	Team members utilize values with a positive intent
Team members are driven by self-interest	Team members are driven by company interest
Team members work in silos	Team members optimize globally
People have territorial behaviors and are quick to perceive suggestions as attacks	People have short toes
People have unhealthy alliances with backroom conversations	People have good intent and actively collaborate with folks
Information is intentionally withheld	Information is shared early (often WIP) and at the same time with all interested parties
People try to undermine each other's credibility by arguing with the weakest part of their argument	People take a "steel man" position and argue against the strongest version of your opponent's position
Folks do not provide direct feedback. Instead, they withhold their thoughts or speak behind each other's backs	Feedback is given directly. This includes feedback about a manager's team
Communicating your own suggestions through a report instead of directly	Feedback is given directly from the person who has it
Evaluating proposals or work by who said or did it instead of by what is in it	Proposals and work is evaluated without regard to who worked on them
Lack of transparency in escalations. Team members go to a manager	Team members speak directly to each other about feedback and requests in order to

Political environment

without first attempting to align with peers on an issue or letting peers know

Non-political environment

resolve their own conflicts. When they escalate, [they do it in an effective way](#)

GitLab CEO Sid Sijbrandij talks about how to prevent politics within a co...



Values make choices

Values make and clarify choices. A well-chosen value has a defensible opposite. Apple, for example, values secrecy over transparency and product perfection over iteration. They are successful building around our counter values — although the result is a very different company.

Values make choices



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, results, and diversity, inclusion & belonging.

Questions 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](#).

New team members should read [GitLab's guide to starting a new remote role](#), and reference [interviews](#) centered on values within the [GitLab Unfiltered YouTube channel](#).

Mission

Our [mission](#) is to **enable everyone to contribute to and co-create the software that powers our world**. This mission guides our path, and we live our values along that path.

Mitigating Concerns

We have a page which documents our [Mitigating Concerns](#). Many of our values help to mitigate some of these concerns.

GitLab Values Quiz

Anyone with a GitLab account can access the GitLab Values Quiz. To participate in the quiz, you will need to complete this [learning course](#) in Level Up. If you have questions, please reach out to our L&D team at learning@gitlab.com.

Last modified June 3, 2025: [Fix broken links \(d7547623\)](#).

[View page source](#) - [Edit this page](#) - please [contribute](#). 