Engineering

The GitLab Product team looks ahead for expanding the platform “What" (customer needs) and “Why” (business strategy) and Engineering determines the “How” (technical implementation) and “When” (scheduling) of the platform releases. The content on this page talks about how we do engineering at GitLab.

Engineering Direction

GitLab has a Three-Year Strategy, and we’re excited to see every member of the Engineering division contribute to achieving it. Whether you’re creating something new or improving something that already exists, we want you to feel empowered to bring your best ideas for influencing the product direction through improved scalability, usability, resilience, and system architectures. And when you feel like you need to expand your knowledge in a particular area, know that you’re supported in having the resources to learn and improve your skills.

Our focus is to make sure that GitLab is enterprise grade in all its abilities and to support the AI efforts required to successfully launch AI features to General Availability.

Making sure that GitLab is enterprise grade involves several teams collaborating on improving our disaster recovery and support offerings through ongoing work with GitLab Dedicated and Cells infrastructure. Our goal here is improved availability and service recovery.

Engineering Culture

Engineering culture at GitLab encompasses the processes, workflows, principles and priorities that all stem from our GitLab Values. All these things continuously strengthen our engineering craftsmanship and allow engineers to achieve engineering excellence, while growing and having a significant, positive impact on the product, people, and the company as a whole. Our engineering culture is primarily being carried and evolves through knowledge sharing and collaboration. Everyone can be part of this process because at GitLab everyone can contribute.

Engineering Excellence

Engineering excellence can be defined as an intrinsic motivation to improve engineering efficiency, software quality, and deliver better results while building software products. Engineering excellence is being fueled by a strong engineering culture combined with a mission: to build better software that allows everyone to contribute.

Engineering Initiatives

Engineering is the primary advocate for the performance, availability, and security of the GitLab project. Product Management prioritizes 60% of engineering time, so everyone in the engineering function should participate in the Product Management prioritization process to ensure that our project stays ahead in these areas. Engineering prioritizes 40% of time on initiatives that improve the product, underlying platform, and foundational technologies we use.

Work in the 40% time budget should be coordinated and prioritized by the Engineering Manager of a team. Use the label `Engineering Time` for issues and MRs that are done as part of it so we can follow the work and the results across the engineering division.

- Contributing to broad engineering initiatives and participating in working group-related tasks.
- Review fixes from our support team. These merge requests are tagged with the `Support Team Contributions` label. You can filter on open MRs.
- Working on high priority issues as a result of issue triaging. This is our commitment to the community and we need to include some capacity to review MRs or work on defects raised by the community.
- Improvements to the performance, stability and scalability of a feature or dependency including underlying infrastructure. Again, the Product team should be involved in the definition of these issues but Engineering may lead here by planning, prioritizing, and coordinating the recommended improvements.
- Improvements and upgrades to our toolchain in order to boost efficiency.
- Codebase improvements: Removing technical debt, updating or replacing outdated dependencies, and enhancing logging and monitoring capabilities.
- Constructing Proof-of-Concept models for thorough exploration of new technologies, enhancements and new possibilites.
- Work on improvements and feature enhancements to the product, in the sense of internal community contributions, that would increase our internal engineering productivity by focusing on ready-to-go items that are currently assigned a low priority in the backlog.

Engineering Innovation

Engineering Innovation is a new process geared toward individual or small-team collaboration that encourages engineers to explore new ideas and Proof-of-Concepts. These projects are typically lean, time-boxed, iterative, and designed to validate whether an idea has the potential to evolve into a viable experimental feature or product. See the Innovation at GitLab Guide.

Technical Roadmaps

Some of the above examples for the 40% time budget can help in forming a long-term technical roadmap for your group, and determine how best to prioritize your technical work to support overall business goals. In addition to the examples above:

- Ask yourself these questions
  - What are your most frequent sources of delays? (Could be long-standing tech debt you have to work past while developing, could be lack of reviewers for your domain, could be external to your team like with pipeline duration)
  - Do you have any consistently similar bugs or security issues that come in due to a certain area?
  - Has your team been talking about potentially refactoring any areas?
  - Is your team struggling with certain processes?
  - Have you had recent incidents that allude to a larger problem?
  - Are you getting frequent requests for help in some area?
  - Is your team frequently missing their deliverable commitments? What would help?
  - Does your area have performance (slow endpoints, inconsistent responses, intermittent errors) or scalability (the feature or area as-is will not scale) concerns?
  - Where do you see the biggest instability? Have you talked to operations and support about feedback for your area?
  - Do you have application or rate limits in the right places?
  - Have you burned down your security, corrective action, and infradev issues?
  - Is your error budget green?
  - Have your feature flags been removed from the codebase yet?
  - Do you have adequate unit test, integration test and E2E coverage?
  - Do you have adequate documentation for your features?
  - Do you have adequate telemetry , logging, monitoring of your features?
  - Do you have adequate error handling and error codes that allows fast and easy diagnostics?
- Gather data like this
  - Master:Broken issues
  - ~"severity::1" and ~"severity::2" bugs

    - Missed-Slo issues
    - Flaky test issues
    - ~“type::maintenance” issues
- Think about the future state of your product
    - Where do you want your product to be this time next year?
    - What are the technical requirements to achieve that?
    - What are technical topics that would benefit from research/POCs?
    - What would make it easier for you to achieve that if it was no longer a factor?
    - What would be the performance and/or business impact once you address these issues?
    - How would you evolve your team processes to regularly review your technical roadmap?

Technical roadmap process

Engineering Managers (EMs) are responsible for collaboratively developing their team’s technical roadmap backlog. All items should be documented as epics and issues using the “Technical Roadmap” label.

Global initiatives will be defined and must be incorporated into each group’s roadmap and prioritization (e.g., allocating 40% of front-end capacity for Vue upgrade, completing all Cells issues for a specific area by milestone XYZ).

Prioritization of items should align with:

1. General business goals
2. Engineering vision
3. Team capacity and expertise

Planning Guidelines:

- Allocate 40% of the overall time budget for technical roadmap items in the normal milestone planning process.
- Use the “Technical roadmap” label for all related issues to facilitate tracking and coordination.

Key Steps:

1. Identify and document technical debt and improvement opportunities
2. Assess impact and effort for each item
3. Prioritize based on business value and strategic alignment
4. Integrate with existing iteration/milestone planning
5. Regularly review and adjust the roadmap

This process ensures a balanced approach between feature development and technical improvements, promoting long-term sustainability and efficiency of the engineering organization.

Community Contributions

We have a 3-year goal of reaching 1,000 monthly contributors as a way to mature new stages, add customer-desired features that aren't on our roadmap, and even translate our product into multiple languages.

Diversity

Diverse teams perform better. They provide a sense of belonging that leads to higher levels of trust, better decision making, and a larger talent pool. They also focus more on facts, process facts more carefully, and are more innovative. By hiring globally and increasing the numbers of women and under represented groups (URGs) in the Engineering division, we're helping everyone bring their best selves to work.

Growing our team

Strategic hiring is a top priority, and we're excited to continue hiring people who are passionate about our product and have the skills to make it the best DevSecOps tool in the market. Our current focus areas include reducing the amount of time between offer and start dates and hiring a diverse team (see above). We're also implementing industry-standard approaches like structured, behavioral, and situational interviewing to help ensure a consistent interview process that helps to identify the best candidate for every role. We're excited to have a recruiting org to partner with as we balance the time that managers spend recruiting against the time they spend investing in their current team members.

Expand customer focus through depth and stability

As expected, a large part of our focus is on improving our product.

For **Enterprise customers**, we're refining our product to meet the levels of security and reliability that customers rightfully demand from SaaS platforms *(SaaS Reliability)*. We're also providing more robust utilization metrics to help them discover features relevant to their own DevOps transformations *(Usage Reporting)* and offering the ability to purchase and manage licenses without spending time contacting Sales or Support *(E-Commerce and Cloud Licensing)*. Lastly, in response to Enterprise customer requests, we're adding features to support Suggested Reviewers, better portfolio management through Work Items, and Audit Events that provide additional visibility into user passive actions.

For **Free Users**, we're becoming more efficient with our open core offering, so that we can continue to support and give back to students, startups, educational institutions, open source projects, GitLab contributors, and nonprofits.

For **Federal Agencies**, we're obtaining FedRAMP certification to strengthen confidence in the security standards required on our SaaS offering. This is a mandated prerequisite for United States federal agencies to use our product.

For **Hosted Customers**, we're supporting feature parity between Self-Managed and GitLab Hosted environments through the Workspace initiative. We're also launching GitLab Dedicated for customers who want the flexibility of cloud with the security and performance of a single-tenant environment.

For customers using **CI/CD**, we're expanding the available types of Runners to include macOS, Linux/Docker, and Windows, and we're autoscaling build agents.

Taking time off

Note: *This process is expected for PTO that is five consecutive days or more, inclusive of adjacent public holidays (excluding weekend days). For PTO that is fewer than five consecutive days, including the cases where there are multiple PTO blocks with fewer than 5 consecutive days and a few working days in-between, a coverage issue is not required but a coverage issue can be filed for PTO of any length, especially if it'd be helpful to balance team continuity and individual flexibility.*

In order to ensure business continuity, and deliver on commitments; the Engineering Division is adopting a PTO Coverage Issue Process. Processes like this are already formalized in GitLab (e.g. PM Coverage Issue) and some team's within Engineering have practiced this regularly at the Management+ level. This allows us to continue to support team member well-being through time away without negatively impacting the rest of the team.

A PTO Coverage issue is required for job grades 8 and up. For job grades 7 and below a PTO Coverage issue is recommended as there is value in going through the process of creating the PTO Coverage issue even if there are minimal items to include (for all levels) in that it forces you to think about what you have on your plate and what impact your PTO will have on those items. So whether the result is that the work waits or there is someone designated as a replacement DRI, it makes the decision explicit and documented.

Once planning for a milestone has been completed (see **Monday, 5 days before the milestone begins**) PTO for periods longer than 5 consecutive days, inclusive of adjacent public holidays (excluding weekend days), cannot be requested. This is to prevent disrupting plans for that milestone. There are exceptions to this, but all need to be discussed with your manager. Examples include:

- urgent scenarios
- a team/individual hits targets earlier in the milestone (we measure impact, not activity)

- a strong need for a team member to take PTO of this length

These issues will help inform teams as they plan their milestones to ensure the work teams are committed to can be achieved with the staff available, or if there will be a lack of staff to achieve those commitments, to work with team members to see what can be done to achieve the results for our customers.

The process below helps to clarify and expand upon the Flexible PTO Policy by making the coordination with the team members manager explicit.

1. Creating an Engineering coverage issue

You should use this issue template to define handshake responsibilities. For extended leave, it is important to find one or more Directly Responsible Individuals (DRIs) that will be able to make decisions while you are away. This may be your manager, another engineer, or maybe the Product Manager for your team. The coverage issue should contain all the necessary information for the DRIs to make good decisions in your absence, so please make sure to include as much detail as needed. The coverage issue should highlight work impact estimates, mitigations identified, and coverage alternatives.

If additional context needs to be shared to provide color to the coverage issue, you can consider a specific handover meeting to cover further details.

It is recommended to work with your manager and other stakeholders when considering cross-functional teammate capacity for a coverage task assignment. For example, while it's optimal for PM, EM, and PDs to assist in covering for each other given their shared knowledge of their product area including customers and users, PM teammates may or may not have the bandwidth or expertise to take on covering engineering specific responsibilities. Alternatively, it may be better for the manager of the engineer or another engineer in the same stage to aid in coverage. Plan to have the necessary conversations across teams and managers.

2. Sharing your Engineering coverage issue with your manager

Once you've filed your engineering coverage issue, share this with your manager prior to milestone planning so they can review and approve. Check the latest guidance in our PTO policy on how much notice is required.

Consider whether any new commitments would be affected by your planned PTO. If a team member falls behind on something, they will need to make sure they have a coverage plan in place to ensure success of their commitments.

3. Manager reviews coverage issue

Once the team member has shared their coverage issue with their manager, the manager will review the coverage issue and validate assumptions with stakeholders or impacted project DRIs as needed.

The manager will make a decision on approval or discuss different arrangements or other contingency plans. Once the manager ticks their box on the coverage issue approving the leave, enter the time off into Workday.

4. Communicate your time off

After team members' coverage issue is approved, team members will communicate their time off and enter the PTO into Deel/Workday including a link to their coverage issue. Team members will share their coverage issue with their relevant colleagues via Slack channels, GitLab status, etc. ahead of the milestone planning.

5. Take your time off

Please disconnect and take the time off that you need!

6. Returning from Time Off

Returning from time off can be overwhelming and daunting. You should work with your DRIs to understand what has changed during your absence and what the current priorities are. Also, communicate transparently that your response time may be slower because you are catching up. Here are some additional tips on how to return back to work after time off.

Engineering Departments

There are five departments within the Engineering Division:

- DevOps Engineering Department
- AI Engineering Department
- Expansion Development Department
- Infrastructure Platforms
- Support Engineering Department

Other Related Pages

- CTO Leadership Team
- Communication
- Database Engineering

- Development Principles
- Engineering Automation
- Engineering Metrics
- Engineering OKRs
- Engineering READMEs
- Frequently Used Projects
- GitLab Innovation Program, managed by the GitLab Legal Team
- Hiring
- Mentorship
- Pajamas Design System
- R&D Tax Credit Applications

Workflows

- Developer onboarding
- Engineering Demo Process
- Engineering Workflow
  - Code Review
  - Security Issues
  - Architecture Design
- GitLab Repositories
- Issue Triage Policies
- Contributing to Go projects
- Wider Community Merge Request Triage Policies
- Root Cause Analysis
- Unplanned Critical Patch releases
- Incident Management

GitLab in Production

- Workflow Diagram
- Error Budgets
- Performance of GitLab
- Monitoring of GitLab.com
- Production Readiness Guide

People Management

- Engineering Career Development

- Engineering Career Mobility Principles
- Emerging Talent @ GitLab
- Engineering Secondments
- Engineering Management
- Starting New Teams

Cross-Functional Prioritization

See the Cross-Functional Prioritization page for more information.

SaaS Availability Weekly Standup

To maintain high availability, Engineering runs a weekly SaaS Availability standup to:

- Review high severity (S1/S2) public facing incidents
- Review important SaaS metrics
- Track progress of Corrective Actions
- Track progress of Feature Change Locks

Infrastructure Items

Each week the Infrastructure team reports on incidents and key metrics. Updating these items at the top of the Engineering Allocation Meeting Agenda is the responsibility of the Engineering Manager for the General Squad in Reliability.

1. Incident Review
    - Include any S1 incidents that have occurred since the previous meeting.
    - Include any incidents that required a status page update.
2. SaaS Metrics Review
    1. Include screenshots of the following graphs in the agenda.
    - Alert Volume Review
    - Corrective Actions

Development Items

For the core and expansion development departments, updates on current status of:

1. Error budgets
2. Reliability issues (infradev)
3. Security issues

Groups under Feature Change Locks should update progress synchronously or asynchronously in the weekly agenda. The intention of this meeting is to communicate progress and to evaluate and prioritise escalations from infrastructure.

Feature Change Locks progress reports should appear in the following format in the weekly agenda:

FCL xxxx - [team name]

- FCL planning issue: `<issue link>`
- Incident Issue: `<issue link>`
- Incident Review Issue: `<issue link>`
- Incident Timeline: `<link to Timeline tab of the Incident issue>`
  - e.g. time to detection, time to initiate/complete rollback (as applicable), time to mitigation
- Cause of Incident
- Mitigation
- Status of Planned/completed work associated with FCL

Feature Change Locks

A Feature Change Lock (FCL) is a process to improve the reliability and availability of GitLab.com. We will enact an FCL anytime there is an S1 or public-facing (status page) S2 incident on GitLab.com (including the License App, CustomersDot, and Versions) determined to be caused by an engineering department change. The team involved should be determined by the author, their line manager, and that manager's other direct reports.

If the incident meets the above criteria, then the manager of the team is responsible for:

- Form the group of engineers working under the FCL. By default, it will be the whole team, but it could be a reduced group if there is not enough work for everyone.
- Plan and execute the FCL.
- Inform their manager (e.g. Senior Manager / Director) that the team will focus efforts towards an FCL.
- Provides updates at the SaaS Availability Weekly Standup.

If the team believes there does not need to be an FCL, approval must be obtained from either the VP of Infrastructure or VP of Development.

Direct reports involved in an active borrow should be included if they were involved in the authorship or review of the change.

The purpose is to foster a sense of ownership and accountability amongst our teams, but this should not challenge our no-blame culture.

Timeline

Rough guidance on timeline is provided here to set expectations and urgency for an FCL. We want to balance moving urgently with doing thoughtful important work to improve reliability. Note that as times shift we can adjust accordingly. The DRI of an FCL should pull in the timeline where possible.

The following bulleted list provides a suggested timeline starting from incident to completion of the FCL. “Business day x” in this case refers to the x business day after the incident.

- Day 0: Incident:
- Business day 1: relevant Engineering Director collaborates with VP of Development and/or VP of Infrastructure or their designee to establish if FCL is required.
- Business day 2: confirmation that an FCL is required for this incident and start planning.
- Business days 3-4: planning time
- Business days 5-9 (1 week): complete planned work
- Business days 10-11: closing ceremony, retrospective and report back to standup

Activities

During the FCL, the team(s) exclusive focus is around reliability work, and any feature type of work in-flight has to be paused or re-assigned. Maintainer duties can still be done during this period and should keep other teams moving forward. Explicitly higher priority work such as security and data loss prevention should continue as well. The team(s) must:

- Create a public slack channel called `#fcl-incident-[number]`, with members
  - The Team’s Manager
  - The Author and their teammates
  - The Product Manager, the stage’s Product leader, and the section’s Product leader
  - All reviewer(s)
  - All maintainers(s)
  - Infrastructure Stable counterpart
  - The chain-of-command from the manager to the VP (Sr Manager, Sr/Director, VP, etc)
- Create an FCL issue in the FCL Project with the information below in the description:
  - Name the issue: `[Group Name] FCL for Incident ####`
  - Links to the incident, original change, and slack channel
  - FCL Timeline
  - List of work items

- Complete the written Incident Review documentation within the Incident Issue as the first priority after the incident is resolved. The Incident Review must include completing all fields in the Incident Review section of the incident issue (see incident issue template). The incident issue should serve as the single source of truth for this information, unless a linked confidential issue is required. Completing it should create a common understanding of the problem space and set a shared direction for the work that needs to be completed.
- See that not only all procedures were followed but also how improvements to procedures could have prevented it
- A work plan referencing all the Issues, Epics, and/or involved MRs must be created and used to identify the scope of work for the FCL. The work plan itself should be an Issue or Epic.
- Daily - add an update comment in your FCL issue or epic using the template:
  - Exec-level summary
    - Target End Date
    - Highlights/lowlights
- Add an agenda item in the SaaS Availability weekly standup and summarize status each week that the FCL remains open.
- Hold a synchronous `closing ceremony` upon completing the FCL to review the retrospectives and celebrate the learnings.
  - All FCL stakeholders and participants shall attend or participate async. Managers of the groups participating in the FCL, including Sr. EMs and Directors should be invited.
  - Agenda includes reviewing FCL retrospective notes and sharing learnings about improving code change quality and reducing risk of availability.
  - Outcome includes handbook and GitLab Docs updates where applicable.

Scope of work during FCL

After the Incident Review is completed, the team(s) focus is on preventing similar problems from recurring and improving detection. This should include, but is not limited to:

- Address immediate corrective actions to prevent incident reoccurrence in the short term
- Introduce changes to reduce incident detection time (improve collected metrics, service level monitoring, which users are impacted)
- Introduce changes to reduce mitigation time (improve rollout process through feature flags, and clean rollbacks)
- Ensure that the incident is reproducible in environments outside of production (Detect issues in staging, increase end-to-end integration test coverage)
- Improve development test coverage to detect problems (Harden unit testing, make it simpler to detect problems during reviews)

- Create issues with general process improvements or asks for other teams

Examples of this work include, but are not limited to:

- Fixing items from the Incident Review which are identified as causal or contributing to the incident.
- Improving observability
- Improving unit test coverage
- Adding integration tests
- Improving service level monitoring
- Improving symmetry of pre-production environments
- Improving the GitLab Performance Tool
- Adding mock data to tests or environments
- Making process improvements
- Populating their backlog with further reliability work
- Security work
- Improve communication and workflows with other teams or counterparts

Any work for the specific team kicked off during this period must be completed, even if it takes longer than the duration of the FCL. Any work directly related to the incident should be kicked off and completed even if the FCL is over. Work paused due to the FCL should be the priority to resume after the FCL is over. Items created for other teams or on a global level don't affect the end of the FCL.

A stable counterpart from Infrastructure will be available to review and consult on the work plan for Development Department FCLs. Infrastructure FCLs will be evaluated by an Infrastructure Director.

Engineering Performance Indicator process

The Product Analytics team is responsible for maintaining Engineering Performance Indicators. Work regarding KPI / RPI is tracked using the Product Analytics task intake tracker.

Manual verification

We manually verify that our code works as expected. Automated test coverage is essential, but manual verification provides a higher level of confidence that features behave as intended and bugs are fixed.

We manually verify issues when they are in the `workflow::verification` state. Generally, after you have manually verified something, you can close the associated issue. See the Product Development Flow to learn more about this issue state.

We manually verify in the staging environment whenever possible. In certain cases we may need to manually verify in the production environment.

If you need to test features that are built for GitLab Ultimate then you can get added to the issue-reproduce group on production and staging environments by asking in the #development Slack channel. These groups are on an Ultimate plan.

Critical Customer Escalations

We follow the below process when existing critical customer escalations requires immediate scheduling of bug fixes or development effort.

Requirements for critical escalation

- Customer is in critical escalation state
- The issues escalated have critical business impact to the customer, determined by Customer Success and Support Engineering leadership
  - Failure to expedite scheduling may have cascading business impact to GitLab
- Approval from a VP from Customer Success AND a Director of Support Engineering are required to expedite scheduling
  - Customer Success: approval from VP, Customer Success Management - Sherrod Patching
  - Support Engineering: approval from VP, Support - Johnny Scarborough

Process

- The issue priority is set to `~"priority::1"` regardless of severity
- The label `~"critical-customer-escalation"` is applied to the issue
- The issue is scheduled within 1 business day
  - For issues of type features, approval from the Product DRI is needed.
- The DRI or their delegate provides daily process updates in the escalated customer slack channel

DRI

- If issue is type bug DRI is the Director of Development
- If issue is type feature DRI is the Director of Product

- If issue requires Infrastructure work the DRI is the Engineering Manager in Infrastructure

The DRI can use the customer critical merge requests process to expedite code review & merge.

Pairing Engineers on priority::1/severity::1 Issues

In most cases, a single engineer and maintainer review are adequate to handle a priority::1/severity::1 issue. However, some issues are highly difficult or complicated. Engineers should treat these issues with a high sense of urgency. For a complicated priority::1/severity::1 issue, multiple engineers should be assigned based on the level of complexity. The issue description should include the team member and their responsibilities.

Team Member	Responsibility
`Team Member 1`	`Reproduce the Problem`
`Team Member 2`	`Audit Code Base for other places where this may occur`

If we have cases where three or five or X people are needed, Engineering Managers should feel the freedom to execute on a plan quickly.

Following this procedure will:

- Decrease the time it takes to resolve priority::1/severity::1 issues
- Allow for a smooth handover of the issue in case of OOO or End of the Work Day
- Provide support for Engineers if they are stuck on a problem
- Provide another set of eyes on topics with high urgency or securing security-related fixes

Internal Engineering handbook

There are some engineering handbook topics that are internal only. These topics can be viewed by GitLab team members in the engineering section of the internal handbook.

AI Engineering

Vision Our goal is not merely to launch features, but to ensure they land successfully and provide ...

Architecture

Complexity at Scale As GitLab grows, through the introduction of new features and improvements on ...

Cross Functional Prioritization

Overview The Cross-Functional Prioritization framework exists to give everyone a voice within the ...

CTO Leadership Team

The CTO Leadership Team is composed of the CTO's direct reports and the Office of the CTO ...

Deployments and Releases

Overview and terminology This page describes the deployment and release approach used to deliver ...

Developer Onboarding

Awesome! You're about to become a GitLab developer! Here you'll find everything you need to start developing.

Development

Development Department Learning and Development

Resources Secure coding best practices It is important that all developers are aware of secure ...

DevOps Engineering

Vision Our goal is not merely to launch features, but to ensure they land successfully and provide ...

Engineering Career Development

The Three Components of Career Development There are three important components of developing ...

Engineering Communication

Communication GitLab Engineering values clear, concise, transparent, asynchronous, and frequent ...

Engineering Demo Process

Occasionally, it is useful to set up a demo on a regular cadence to ensure cross-functional ...

Engineering Error Budgets

The error budget provides a clear, objective metric that determines how unreliable the service is allowed to be within a single quarter.

Engineering Function Performance Indicators

Executive Summary KPI Health Status Engineering Handbook MR Rate Okay Above target Engineering Team ...

Engineering Hiring

Overview Hiring is a cornerstone of success for our engineering organization, contributing to our ...

Engineering IC Leadership

Engineering IC Leadership at GitLab: going beyond Senior level At GitLab, it is expected that ...

Engineering Management

How Engineering Management Works at GitLab At GitLab, we promote two paths for leadership in ...

Engineering Mentorship

Mentorship, Coaching and Engineering Programs Senior Leaders in Engineering The 7CTOs Program is run ...

Engineering Projects

Name Location about.gitlab.com gitlab-com/marketing/digital-experience/about-gitlab-com AI Gateway ...

Engineering Secondments

Learn about GitLab's secondment program for external engineers.

Engineering Team Readmes

Engineering Workflow

This document explains the workflow for anyone working with issues in GitLab Inc.

Expansion Development Department

Vision Scale and develop our diverse, global team to drive results that support our product and ...

Fast Boot

A Fast Boot is an event that gathers the members of a team or group in one physical location to work ...

Frontend Group

Teams Create Monitor Plan Secure Verify and Release Frontend domain experts You can find engineers ...

GitLab Repositories

GitLab consists of many subprojects. A curated list of GitLab projects can be found at the GitLab ...

Guidelines for automation and access tokens

Guidelines for automation with project/group tokens or service accounts

Incident

Definition of an Incident The definition of “incident” can vary widely among companies ...

Infrastructure

The Infrastructure Department is responsible for the availability, reliability, performance, and scalability of GitLab.com and other supporting services

Infrastructure Platforms

The Infrastructure Platforms department is responsible for the availability, reliability, performance, and scalability of GitLab SaaS Platforms and supporting services

Innovation at GitLab

This guide serves as a comprehensive handbook for GitLab team members (engineers, product managers, ...

Joint R&D OKR Process

Monitor Stage

The Monitor Stage is responsible providing observability and response features.

Monitoring of GitLab.com

GitLab.com Service Availability The calculation methodology for GitLab.com Service Availability ...

On-Call

If you're a GitLab team member and are looking to alert Reliability Engineering about an ...

Open Source at GitLab

We believe in Open Source As a company, GitLab is dedicated to open source. Not only do we believe ...

Performance

Performance Facets We categorize performance into 3 facets Backend Frontend Infrastructure Backend ...

Policies related to GitLab.com

The handbook pages nested under "policies" directory are controlled documents, and ...

R&D Tax Credits

GitLab submits applications for R&D Tax Credits in a number of jurisdictions that implement ...

Recognition in Engineering

Engineering Quarterly Achievers Quarterly, CTO Leadership will recognize Engineering team members ...

Releases

Overview and terminology This page describes the processes used to release packages to self-managed ...

Root Cause Analysis

At GitLab transparency is one of our core values, as it helps create an open and honest working ...

Starting new teams

Starting new teams Our product offering is growing rapidly. Occasionally we start new teams. Backend ...

Testing

Welcome to the Testing Guide. Pages in this section provides information about testing practices, ...

Unplanned Upgrade Stop Workflow

An unplanned upgrade stop is disruptive for customers as it requires to perform rollback and ...

Last modified May 26, 2025: Separate DevOps and AI Engineering, deprecate Development (8a4a60ac)

View page source - Edit this page - please contribute.