IN1729

INSEAD The Business School for the World®

Case Study

GitLab: Can “All Remote” Scale?

12/2020-6579

This case study was written by Marco Minervini, Post-Doctoral Fellow, with assistance from Jean Wee, Research Associate, under the supervision of Phanish Puranam, Professor of Strategy and the Roland Berger Chaired Professor of Strategy and Organisation Design, all at INSEAD. It is intended to be used as a basis for class discussion rather than to illustrate either effective or ineffective handling of an administrative situation.

To access INSEAD teaching materials, go to https://publishing.insead.edu/case/gitlab

Introduction

GitLab, a start-up in the booming software development tools industry, was an “all remote” company. All its employees – more than 1,000 in 59 countries– worked remotely, and typically asynchronously, often without ever coming into physical contact. With no headquarters or offices, they were free to work wherever and whenever they chose. But the company was soon to go public, which made it important to assess the scalability of the all-remote approach, not least because the workforce had almost tripled in 2019. How would such a radically distributed organisation work on a large scale? Could it be easily replicated (by competitors)? What lesson could learned from GitLab by from organizations struggling to go remote in response to the COVID-19 pandemic? What were the implications of “all remote” for the future of work?

Background

GitLab developed tools that allowed software engineers to automate many parts of the software development cycle – from initial planning to the final deployment and monitoring of new code in use.[1] It was widely recognized for its “continuous integration” (CI) product,[2] which enabled teams of coders to cut a complex project into chunks, work in parallel on specialized tasks, then put the pieces back together again into a functioning whole. Specifically, GitLab’s CI tools automated the verification of the compatibility of new contributed code added to the existing code base – in effect representing the automation of the coordination work previously conducted by a human coder.

The basic GitLab product was created in 2011 by Dmitriy Zaporozhets, drawing on the open source version control system[3] known as Git (made by Linus Torvalds). He was joined by Sid Sijbrandij as CEO and the company was incorporated in 2014.[4] Since then, more than 100,000 client organizations around the world, and spanning different sectors, had used GitLab, including Goldman Sachs, NASA, the University of Washington, and the US Airforce. As of October 2019, GitLab had over 1,000 employees[5] and 150 open positions,[6] a significant expansion from the 374 headcount a year earlier.[7]

Apart from its extraordinary growth, GitLab was noteworthy for at least three other reasons. First, like many technology companies, it used its own tools – GitLab (the company) used GitLab (the product) to make improvements to GitLab (the product). And it used those same tools to organize and manage itself. For example, the company handbook,[8] which exhaustively documented its formal organizational structure and processes, was itself developed, maintained and edited as if it were a code repository. Second, the handbook was public; anybody could view it from inside or

1 https://about.gitlab.com/stages-devops-lifecycle/
2 https://www.forrester.com/report/The+Forrester+Wave+CloudNative+Continuous+Integration+Tools+Q3+2019/-/E-RES148217#
3 Version control systems were a category of software tools that helped a software team manage changes to source code over time. Version control software kept track of every modification to the code in the code repository. Git was one of the most popular version control systems in use.
4 For a brief company history: https://about.gitlab.com/company/history/
5 https://about.gitlab.com/company/team/
6 https://about.gitlab.com/jobs/apply/
7 https://web.archive.org/web/20181030162842/https://about.gitlab.com/company/team/
8 https://about.gitlab.com/handbook/

outside the company. Third, it was one of the largest “all remote” commercial organizations in the world.

“All Remote” by Design

In response to a Gallup survey of 2017, 47% of workers in America claimed to work remotely to some extent. GitLab was an extreme case. There were no headquarters (except for postal and legal purposes) – it was “all remote” by design. Employees could work anywhere they chose, which applied to all levels of the hierarchy – even C-level managers. Since they were spread across almost every time zone, there were no predetermined working hours – they could choose the time they preferred. Even hiring, onboarding and firing happened remotely.

The only instance where all GitLab employees were co-located was at the yearly “Contribute” event, a mainly informal corporate get-together for team members to “get face time with one another, build community and reinforce cultural values”.[9]

The Competition

GitLab was not alone in offering integrated functionalities to automate the entire lifecycle of software development. The competition was fierce and diverse.

Market leader GitHub likewise relied on Git design principles and had functionalities that supported several phases of the software development lifecycle. In 2018 it was acquired by Microsoft for $7.5 billion. Microsoft was a player in its own right, with its own CI tools even before the acquisition. So were other global software majors such as Google and Amazon. Atlassian, a software company listed on the NASDAQ, had grown through acquisitions in the last 10 years, including Bitbucket (offering a CI tool), Trello (a project management tool) and Jiira (an issue and project tracking tool). GitLab also competed with other start-ups in various phases of maturity (e.g., Plutora, Asana), as well as open source software communities like Jenkins CI, and Redmine.[10]

The GitLab Way: The First Few Weeks from a New Hire’s Perspective

> *“We don't expect you to hit the ground running from day one. We highly recommend taking at least two full weeks for onboarding and only in week three, starting with team specific onboarding and training.”*[11]

On his first day, Victor had a welcome call with Mariel, his manager. Throughout his time at in GitLab, they would have a scheduled weekly one-to-one call. Like every new team member,[12] Victor received a link to an onboarding checklist[13] and was assigned a buddy – another employee

9 https://about.gitlab.com/company/culture/contribute/; Participation to the contribute event is not mandatory for GitLab employees, but it is highly encouraged. Latest contribute events had an attendance of about 85% of GitLab employees.
10 For a full map of competing tools in every stage of software development see https://about.gitlab.com/devops-tools/
11 https://about.gitlab.com/handbook/general-onboarding/
12 Team Member is a term used internally by GitLab to call its employees. The two terms will be used interchangeably through the case.
13 https://gitlab.com/gitlab-com/people-group/employment-templates/-/blob/master/.gitlab/issue_templates/onboarding.md

in the same time zone with at least three months' tenure. The onboarding checklist had two sections: one common to all employees, the other department- and role-specific.

As he had been told during the interview process: "Your first week is not about your job, it is about learning how to work in the GitLab way." Besides administrative tasks like setting up a profile and accounts for the tools used on the job, Victor was asked to carefully read the GitLab handbook and set up at least five 30-minute calls with other employees, preferably in other departments and other time zones. Even after the onboarding period, he was encouraged to dedicate a few hours a week to informal calls with colleagues.

There were several other occasions for informal interactions. Every Tuesday and Thursday he could attend *Take a Break Calls*, where members of GitLab talked about non-work-related subjects like their hobbies and interests. To kick off the conversation, team members could choose between five predetermined topics, which were updated each time. In addition, since Victor had a passion for gardening, whenever he wanted a break with fellow employees who shared his passion for gardening, he could chat on the dedicated #gardening Slack channel.

As part of the onboarding process, Victor was given a set of tasks to familiarize himself with the GitLab approach. To be conversant with Git-based workflows[14] he had to go into the shared repository containing the team.yml file, which stored information on every team member. This was the source of information for a web page displaying the organizational chart (see Exhibit 1) – his first task was to add himself to the page. This simple task captured the core philosophy of "Git-based" work processes. To modify the page (see Exhibit 2), he needed to:

- "fork" the team.yml file (generate a copy of it)
- modify it (adding his name, job title, location, and to whom he reported)
- make a "merge request" (ask a maintainer – a person in GitLab in charge of ensuring the quality of the file to substitute the old file with the new one)

After an automated technical check to ensure there was no technical incompatibility in the way information was added, the maintainer would decide whether to approve the request. This was the Git process to follow for every change made to the code base – the repository of code that comprised the GitLab product – and in fact also to the company handbook.

As stated in the handbook, "Everything starts with a merge request". Whenever possible, Victor was expected to directly act on the codebase or on the handbook. For any change in the codebase, the starting point was to propose a direct change to the code; a discussion would follow from that. The same applied to the handbook. For instance, if his manager Mariel wanted to create a new job family, she first had to send a merge request to add the job description to the handbook. The new job family was ultimately approved by Sijbrandij, the CEO.

After checking that Victor had completed his first week of onboarding tasks, in their weekly one-to-one Mariel suggested a couple of pending ones he could start working on. He was free to choose among them. While to Victor they seemed particularly simple, Mariel had picked them intentionally. In her experience, engineers tended to choose very complex tasks at the beginning

14 A good reference on how Git works: https://about.gitlab.com/blog/2019/08/02/gitlab-for-the-non-technical/

to showcase their ability, but the priority was to become familiar with the GitLab way of working. The tasks, while simple, would help him get used to GitLab culture.

He was repeatedly reminded that communication was deliberately biased toward "asynchronous and public" and – to keep things transparent – he should favour communications that left a trace and were visible to the broadest number of other employees, and he should not necessarily expect a real-time "chat style" interaction. For instance, for the first assignment, when he had needed information to complete the task he had sent an email query to Mariel. She replied kindly but firmly that, as indicated in the communication section of the handbook, "We do not send internal email here." Instead, she suggested he put up the question on the Slack channel of the team. He later got a precise answer from one of his peers. Then, as team leader, it was up to Mariel to decide whether that information needed to last (i.e., remain permanently visible). If so, it would be stored in a place where the information could last and be available to everyone in the company, in an "issue" document or on a page in the handbook. These were the ground rules for internal communications – the means that enable GitLab employees to work asynchronously and remotely.

While exploring the company handbook, Victor realized that a technical page required updating. It was lacking in clarity and did not represent how GitLab was currently using that technology. Team members were encouraged to make improvements to the handbook and in their weekly one-to-one, Mariel agreed that the suggested update was much needed. Following the Git process, Victor forked the page of the handbook and carefully worked to modify it until its full completion. It took a long time, but he wanted to complete all the needed changes before sending in the merge request to integrate them into the handbook.

However, to his surprise, Joanne, a member of another team relying on the same technology, had already taken this in hand by the time he submitted the merge request. Not knowing what Victor was working on, she had already changed the document – and some of her changes were incompatible with Victor's contribution. It was a lesson: Victor needed to promptly submit any Minimum Viable Change so that others would know that he was working on a section of the codebase (or the handbook) and could then build on his changes. Indeed iteration was one of GitLab's core values: "We do the smallest thing possible and get it out as quickly as possible."[15]

Structuring Work at GitLab

Natalia, a manager in People Operations, was responsible for "triaging" (i.e. distributing and allocating) tasks in her team. Inputs came from all across the company – everyone in GitLab (including users) could report a problem to Natalia if it was related to her team. She would check the issues reported, prioritize them, and decompose them into tasks. If unsure of what needed to be done, she might do some research. Martin, a fellow manager, did it differently. He would delegate the research process to one of his direct reports, and only then determine which tasks needed to be performed. Charlize, head of a creative team, did things yet another way. She would lay out the issue on a shared document and let the team contribute their views to develop a better understanding of what needed to be done, and only then divide the problem into tasks. After clarifying the necessary tasks, Natalia would decompose them to maximize their discreteness and

15 https://about.gitlab.com/handbook/values/#iteration; https://about.gitlab.com/handbook/values/#make-small-merge-requests

independence from each other and assign a priority level to each of them. Tasks would then be displayed on the team's electronic dashboard to which each team member had access. Natalia mostly left her reports free to choose which task to perform.

For example, John, one of her reports, could assign himself to one of the tasks so that the others would know that he was responsible. From then on, he was in charge of completing that task. If the task referred to the codebase or the handbook, he would need to make a merge request that contained the proposed change, to be approved by a maintainer ("the person who knows most about the code base you are changing") selected from a list of authorized team members. The maintainer would decide whether to accept, require some changes to the merge request, or reject it.

Remote, Not Flat

> *"We are not a democratic or consensus-driven company. People are encouraged to give their comments and opinions, but in the end one person [the Directly Responsible Individual] decides the matter after they have listened to all the feedback."*[16]

Anita was the "Directly Responsible Individual (DRI)" – appointed by her manager – in charge of improving the speed of visualization of a data dashboard. Being a DRI meant she was responsible for completion of the task. If she needed help, she could search for it, but she retained full responsibility for accurate completion and was also the ultimate decision maker on how to proceed next. Before starting, the task had to be articulated in a document known as an "issue", explaining why the specific task was needed and how she planned to proceed. This information was public so that everyone in GitLab (both employees and external contributors) could comment on the issue and suggest alternative approaches.

Having outlined the issue and the solution, Anita solicited feedback from other members of the organization whom she believed could provide inputs on what to do next. One of them, Ankit, suggested that another code structure could be more effective (his team had switched from the one Anita was proposing some months earlier). Ankit collected all the evidence available on this past project to support his idea, which indicated that his solution would increase visualization speed by 20%.

Working asynchronously allowed Anita to carefully go through Ankit's suggestion without time pressure to reply, and work through the pros and cons. While she still had doubts whether his solution could generate the same benefits in this specific code structure, the evidence he provided seemed solid. Once she decided to try out Ankit's proposed solution, she could check whether the performance gain would still be present given the specific code structure she was facing. Ultimately, as the DRI, she could easily reverse the solution. If his suggestion proved ineffective, she could revert to her own initial strategy.

Transparency in Compensation

16 https://about.gitlab.com/handbook/leadership/#gitlab-team-members

Salary levels were determined by a compensation calculator – made public – based on the role, the level of expertise within that role, and a location factor. Salary was computed using the San Francisco job market as a reference point and adapted to the cost of labour of the city where the GitLab employee lived.

For example, Jamal lived in Paris, where the location factor was 0.6 – hence his salary was 60% of an employee with the same role and experience in San Francisco. He also had stock options, which were assigned to everyone, according to role, irrespective of location. Each role came with a pre-determined number of stock options.

In the daily all-company meeting, Mary, his manager, announced that Jamal had been awarded two discretionary bonuses. A discretionary bonus of $1,000 was awarded whenever a GitLab employee behaved in an exemplary way in accordance with company values (see Exhibit 3). There was no limit to the number of discretionary bonuses they might receive. In Jamal's case, both Annie and Mario had recommended a discretionary bonus for two different projects he was working on. In GitLab, anyone could nominate another member of the organization for a discretionary bonus. Annie simply had to send a message to Jamal's manager with a short description of what he had done and how it reflected GitLab's values. Annie cited his "collaboration" – he was very supportive, had worked constructively on the suggestions received on his merge request, and accurately documented his actions. For Mario, Jamal perfectly embodied the value of "iteration" – by sharing any minimum viable change he made, Jamal had greatly facilitated task of shipping the code on time.

Scale-up?

In September 2019, after its Series E round of funding, GitLab was valued at $2.7 billion.

Now that the company was gearing up to go public, many observers (and indeed company insiders) wondered if its "all remote" high-transparency model would be viable for a publicly listed company. GitLab relied extensively on transparency within and outside the company. Transparency was a core value and fundamental to internal coordination and coordination with external contributors and clients alike. Thanks to the fact that "Everything [was] public by default", anyone could see its internal functioning.[17]

Sidbrandij argued that in a world of rising national boundaries, visa restrictions, and limited talent pools in any one location, GitLab's model was a great advantage: "Access to most of the world talent and no wasted commuting time!"

Moreover, the emergence of the COVID-19 pandemic in early 2020 had seen organizations forced to adopt remote working as part of social distancing measures to mitigate the spread of virus. GitLab now seemed more like a guidepost than an oddity.

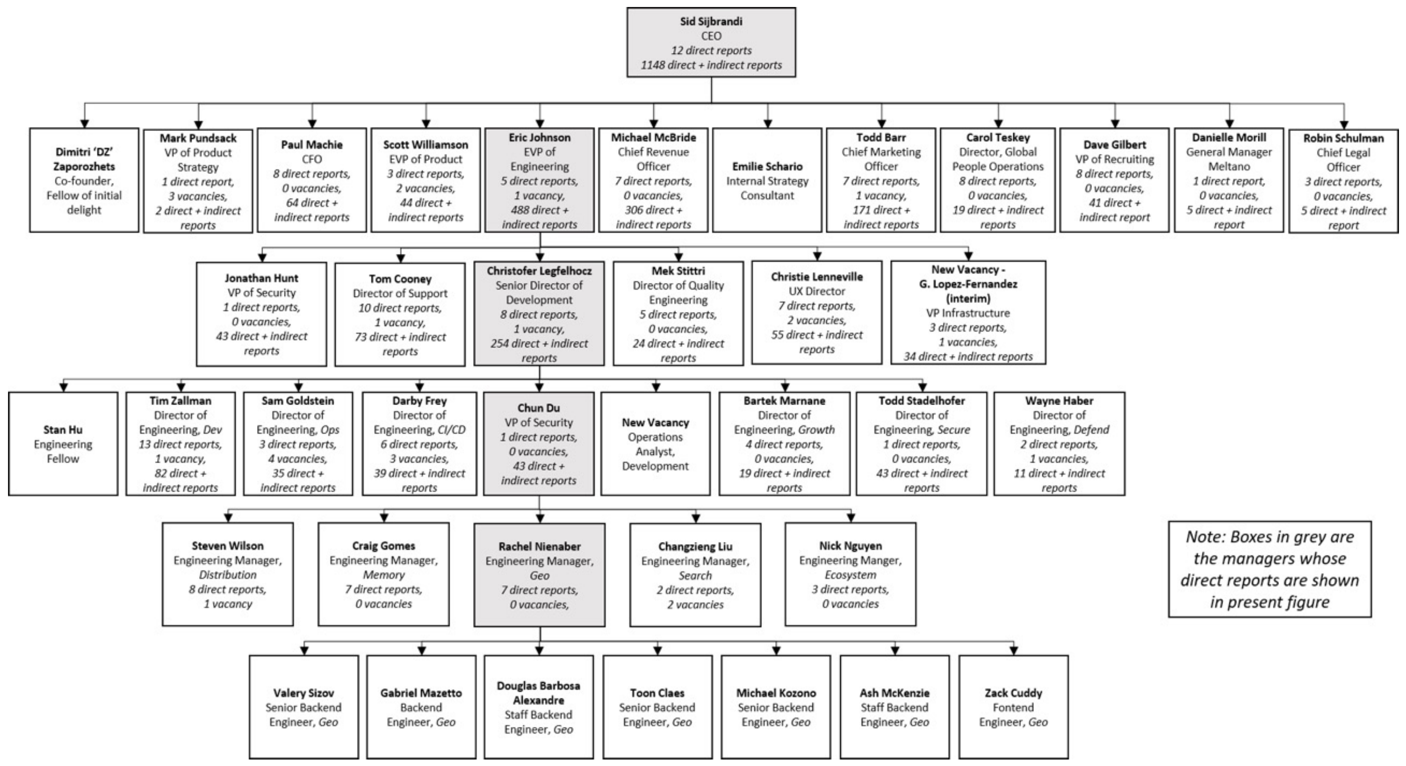
Could the core ingredients that ensured GitLab's ability to thrive through an "all remote" organizational design withstand the pressure of a sudden expansion of the workforce and stronger

17 https://about.gitlab.com/handbook/values/#public-by-default. Not all information on GitLab was public. Public by default means that category of information was public unless there was a reason for it not to be. For a list of the type of information that were not public, see: https://about.gitlab.com/handbook/values/#not-public

market scrutiny and competition? What could organizations in other sectors learn from its approach?

Exhibit 1

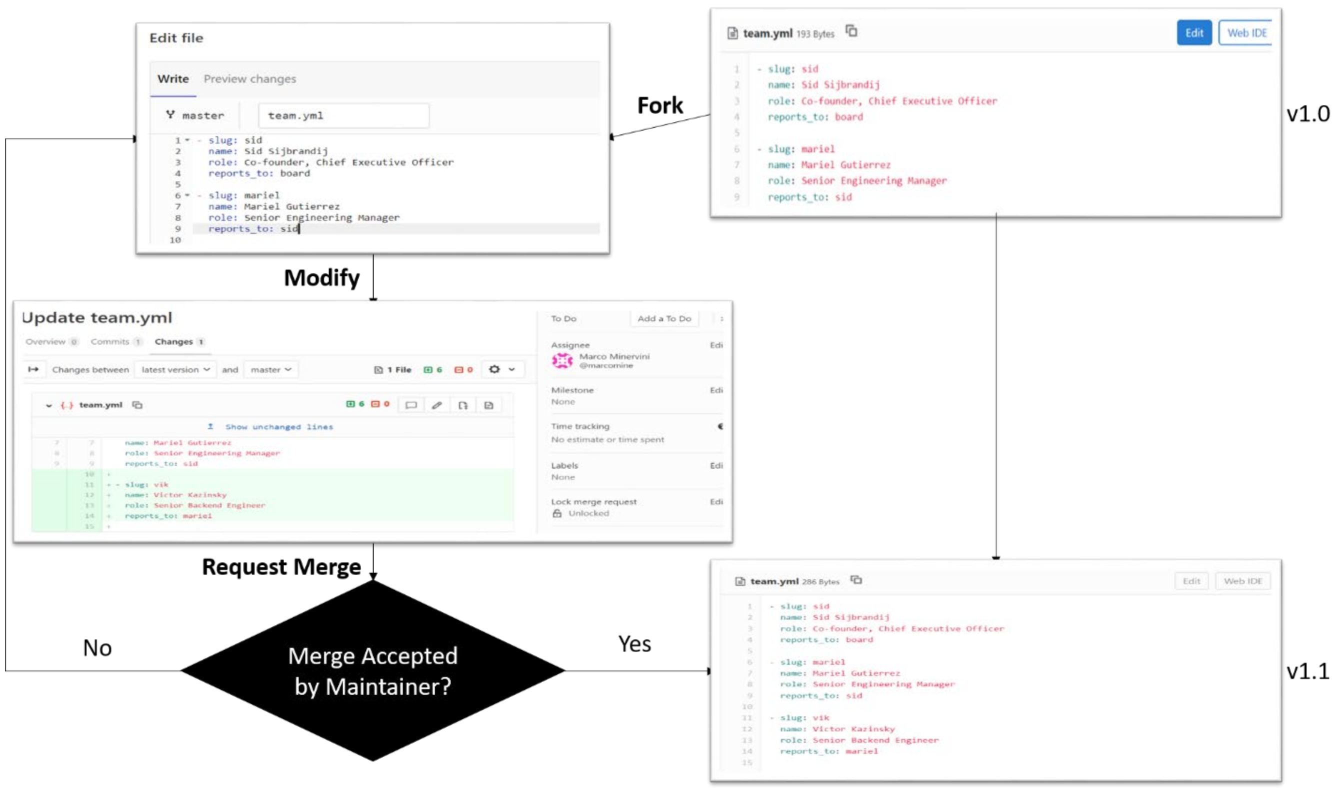
Section of GitLab Organizational Chart (as of 4 February 2020)


Sid Sijbrandi CEO 12 direct reports 1148 direct + indirect reports
Dimitri 'DZ' Zaporozhets Co-founder, Fellow of initial delight
Mark Pundsack VP of Product Strategy 1 direct report, 3 vacancies, 2 direct + indirect reports
Paul Machie CFO 8 direct reports, 0 vacancies, 64 direct + indirect reports
Scott Williamson EVP of Product 3 direct reports, 2 vacancies, 44 direct + indirect reports
Eric Johnson EVP of Engineering 5 direct reports, 1 vacancy, 488 direct + indirect reports
Michael McBride Chief Revenue Officer 7 direct reports, 0 vacancies, 306 direct + indirect reports
Emilie Schario Internal Strategy Consultant
Todd Barr Chief Marketing Officer 7 direct reports, 1 vacancy, 171 direct + indirect reports
Carol Teskey Director, Global People Operations 8 direct reports, 0 vacancies, 19 direct + indirect reports
Dave Gilbert VP of Recruiting 8 direct reports, 0 vacancies, 41 direct + indirect report
Danielle Morill General Manager Meltano 1 direct report, 0 vacancies, 5 direct + indirect report
Robin Schulman Chief Legal Officer 3 direct reports, 0 vacancies, 5 direct + indirect report
Jonathan Hunt VP of Security 1 direct reports, 0 vacancies, 43 direct + indirect reports
Tom Cooney Director of Support 10 direct reports, 1 vacancy, 73 direct + indirect reports
Christofer Legfelhocz Senior Director of Development 8 direct reports, 1 vacancy, 254 direct + indirect reports
Mek Stittri Director of Quality Engineering 5 direct reports, 0 vacancies, 24 direct + indirect reports
Christie Lenneville UX Director 7 direct reports, 2 vacancies, 55 direct + indirect reports
New Vacancy - G. Lopez-Fernandez (interim) VP Infrastructure 3 direct reports, 1 vacancies, 34 direct + indirect reports
Stan Hu Engineering Fellow
Tim Zallman Director of Engineering, Dev 13 direct reports, 1 vacancy, 82 direct + indirect reports
Sam Goldstein Director of Engineering, Ops 3 direct reports, 4 vacancies, 35 direct + indirect reports
Darby Frey Director of Engineering, CI/CD 6 direct reports, 3 vacancies, 39 direct + indirect reports
Chun Du VP of Security 1 direct reports, 0 vacancies, 43 direct + indirect reports
New Vacancy Operations Analyst, Development
Bartek Marnane Director of Engineering, Growth 4 direct reports, 0 vacancies, 19 direct + indirect reports
Todd Stadelhofer Director of Engineering, Secure 1 direct reports, 0 vacancies, 43 direct + indirect reports
Wayne Haber Director of Engineering, Defend 2 direct reports, 1 vacancies, 11 direct + indirect reports
Steven Wilson Engineering Manager, Distribution 8 direct reports, 1 vacancy
Craig Gomes Engineering Manager, Memory 7 direct reports, 0 vacancies
Rachel Nienaber Engineering Manager, Geo 7 direct reports, 0 vacancies,
Changzieng Liu Engineering Manager, Search 2 direct reports, 2 vacancies
Nick Nguyen Engineering Manger, Ecosystem 3 direct reports, 0 vacancies
Valery Sizov Senior Backend Engineer, Geo
Gabriel Mazetto Backend Engineer, Geo
Douglas Barbosa Alexandre Staff Backend Engineer, Geo
Toon Claes Senior Backend Engineer, Geo
Michael Kozono Senior Backend Engineer, Geo
Ash McKenzie Staff Backend Engineer, Geo
Zack Cuddy Fontend Engineer, Geo
Note: Boxes in grey are the managers whose direct reports are shown in present figure

Source: Adapted version of GitLab Organizational Chart (https://about.gitlab.com/company/team/org-chart/)

Exhibit 2

The Git Process Followed by Victor to Include Himself in team.yml



```
- slug: sid
  name: Sid Sijbrandij
  role: Co-founder, Chief Executive Officer
  reports_to: board

- slug: mariel
  name: Mariel Gutierrez
  role: Senior Engineering Manager
  reports_to: sid
```

```
- slug: sid
  name: Sid Sijbrandij
  role: Co-founder, Chief Executive Officer
  reports_to: board

- slug: mariel
  name: Mariel Gutierrez
  role: Senior Engineering Manager
  reports_to: sid
```

```
  name: Mariel Gutierrez
  role: Senior Engineering Manager
  reports_to: sid
+
+ - slug: vik
+   name: Victor Kazinsky
+   role: Senior Backend Engineer
+   reports_to: mariel
+
```

```
- slug: sid
  name: Sid Sijbrandij
  role: Co-founder, Chief Executive Officer
  reports_to: board

- slug: mariel
  name: Mariel Gutierrez
  role: Senior Engineering Manager
  reports_to: sid

- slug: vik
  name: Victor Kazinsky
  role: Senior Backend Engineer
  reports_to: mariel
```

Source: Authors

Exhibit 3

GitLab Values and Links to their Description in Company Handbook

GitLab Values	
Collaboration	https://about.gitlab.com/handbook/values/#collaboration
Results	https://about.gitlab.com/handbook/values/#results
Efficiency	https://about.gitlab.com/handbook/values/#efficiency
Diversity	https://about.gitlab .com/handbook/values/#diversity
Inclusion	https://about.gitlab.com/handbook/values/#inclusion
Iteration	https://about.gitlab.com/handbook/values/#iteration
Transparency	https://about.gitlab.com/handbook/values/#transparency