PLANNING FOR SUCCESS

HOW TO QUICKLY ITERATE AND SHIP CODE IN THE DEVOPS LIFECYCLE

WHAT'S INSIDE?

INTRODUCTION

Software touches nearly every aspect of daily life. Reducing the time between thinking of an idea and having the code in production is vital to providing value to customers. Planning what actions to take to ship an idea is an essential part of maintaining velocity, and organizations must find ways to simplify processes to empower teams to focus on delivering value to customers faster. When teams aren't plodding through tangled planning processes, they can quickly ship code and innovate to create features that customers want.

The planning stage of the software development lifecycle is one of the most crucial components because it builds the foundation for workflows, project management, and resource allocation. Development and operations teams must have a strong collaborative relationship during this stage in order to maintain progress and meet business objectives. The planning stage has historically been a source of confusion and difficulty, with 25% of product managers struggling to create a consistent project management methodology or scoping document, according to The State of Project Management Survey. To improve their ability to deliver alongside a product roadmap, teams have undergone a series of transformations.

Waterfall method

The Waterfall model was the industry's first attempt at project management. In this linear approach, teams complete sequential steps in software development and agree on delivery details early in the lifecycle. By investing time in the beginning to outline requirements, planning and designing becomes straightforward and progress can be easily measured. While some teams appreciated the structured approach to project management, others looked for a more flexible approach and turned to an Agile methodology.

Agile method

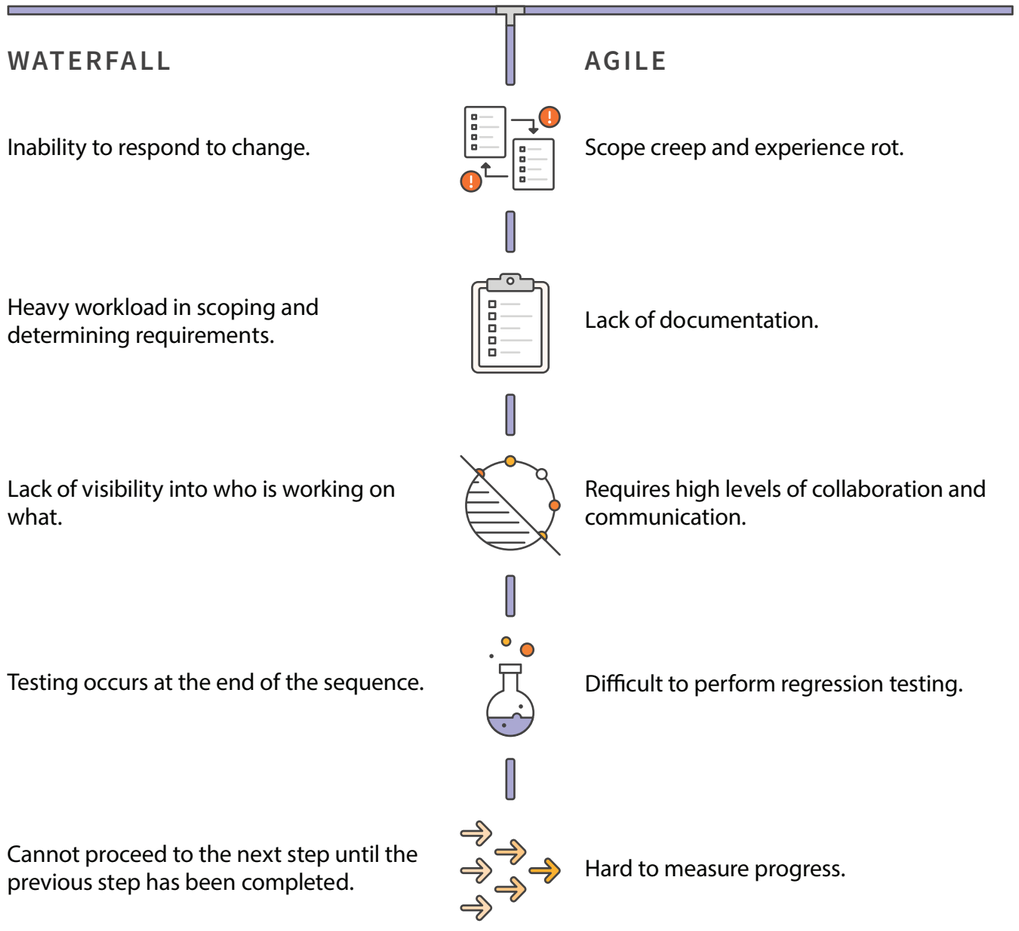
The Agile model empowers teams to rapidly respond to changes and cultivates a mindset that embraces "collaboration, a positive outlook, an openness to change, a willingness to fail (and recover fast), and transparency." In this iterative approach, time is allocated into "sprints" with a list of deliverables, ensuring rapid delivery. Teams can develop a minimum viable product and improve its features in successive iterations. Agile project planning paves the way for multiple changes throughout a project, but because teams work in sprints, some features may not be delivered within the allotted time, requiring additional sprints, which prolongs the software development lifecycle.

These traditional ways to plan helped teams organize and determine the scope of projects, but they weren't fully meeting the needs of modern teams.

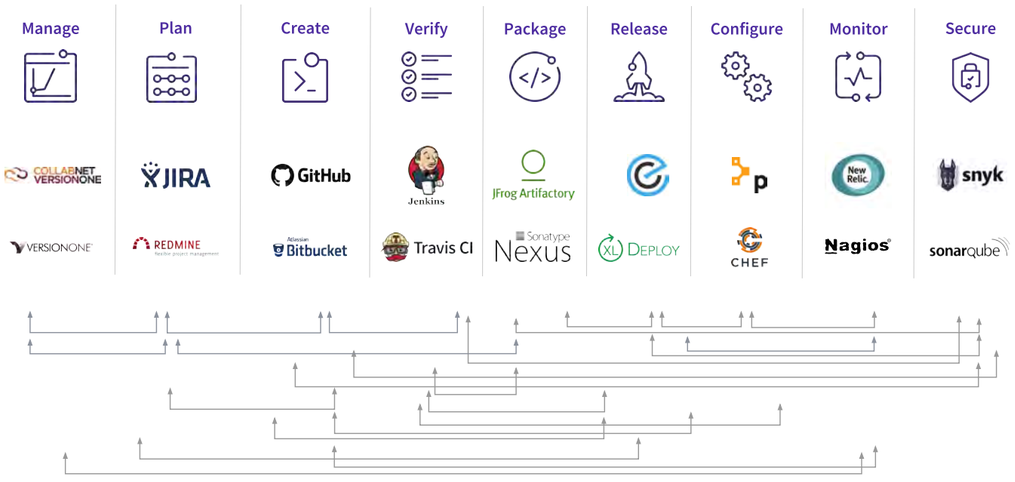
THE PROBLEM WITH TRADITIONAL PLANNING METHODS

Traditional workflows, such as Waterfall and Agile, usually involve multiple teams using a variety of methods and tools, including word processing, spreadsheets, project management tools, and email and chat communications, to plan work and iterate on ideas. Some teams may adopt specific tools that are not widely used within an organization, while other teams prefer to have in-person meetings to plan work.

Team members shuffle from one meeting to the next, attending daily stand-ups and weekly sync-up meetings with peers to track progress and collaborate. In addition to collaboration meetings, status meetings occur to update external stakeholders, such as IT leadership or members of other teams, which require reports that include visuals, like graphs and charts, to illustrate progress. In between all these meetings, team members document every conversation to record discussions and decisions. These additional artifacts, as part of every meeting, must be stored somewhere, requiring an additional tool which may not be easily accessible to everyone on every team.



HOW A LARGE TOOLSET LIMITS PROGRESS



Integration complexity of toolchains slows down teams.

The idea behind using a variety of tools is understandable: Teams can take advantage of the best software in the industry. But, having such a large toolset hinders progress rather than enables it, causing longer cycle times, a lack of visibility, and the creation of silos. These silos take overhead to integrate and manage, slowing down teams and preventing collaboration and communication.

With all the meetings involved in aligning work, there's a lot of communication overhead. Team members who communicate with one person don't know whether the second or third person or rest of the team has access to conversation. In a Waterfall or Agile workflow, this is one of the greatest challenges that a team faces: Making a decision or having a private discussion means that other team members are excluded by default. If team members are included after the initial contact, it requires a restart of the conversation and additional context to align everyone. If information is accidentally omitted from the conversation, team members may not have a full picture of requirements, preventing them from effectively executing.

Compounding the problem with meetings and multiple in-person conversations is the challenge of developing and accessing the right information. There is no clear place where a single source of truth exists, so team members have to access different tools to aggregate conversations, and they may not know the latest decision or the most up-to-date progress report. When team members are forced to hunt for information, they are at risk of missing half the conversation or developing decisions based on outdated information, putting a project at risk.

Further jeopardizing velocity is the friction people—especially new hires—face before contributing to projects. The current toolchain crisis means the high number of tools

and the various types of tools force people to troubleshoot new programs and acquire new skills for each phase of the software development lifecycle. It's not enough to master a few well-known tools; people must dedicate time to learning new software.

When decisions are made in a variety of tools with a limited number of stakeholders, collaboration suffers. Traditional settings require people to be in the same space and time, disabling asynchronous collaboration. Teams don't have flexibility and are pressured with time limitations, tasked with juggling their daily and weekly status meetings - with peers and managers - and additional responsibilities with collaborative ideation work. There is little time left for collaboration when working in silos and constantly playing catch up on information.

A need for organization and visibility to accelerate delivery velocity

Managing projects and software is inherently complex, but the tools don't have to be. Regardless of how many projects, people, or products a team manages, getting visibility to keep things running smoothly should be the easy part.

Collaboration is driven by conversation. It should be a natural and integrated practice throughout the development lifecycle, not a manual, contrived process that requires administration and maintenance. With the growing number of necessary collaboration tools to ship software faster, modern development teams need to support cross-functional collaboration at scale to ensure delivery velocity.

Rapid delivery requires reducing every change proposal to its absolutely minimally viable form to allow teams to ship almost anything within a single release, obtain immediate feedback, and avoid deep investments in ideas that might not work. A minimum viable change also prevents over-engineering and encourages teams to quickly iterate, helping teams consider the full scale of development complexity but focus on moving a project forward with the simplest solution. Traditional workflows, like Waterfall and Agile, don't easily facilitate minimum viable changes, slowing down delivery velocity and preventing businesses from shipping the features that customers want.

Quickly delivering what customers want requires a modernized software development lifecycle that saves time, effort, and cost. Continuous integration and continuous delivery (CI/CD) is the cornerstone of modern software development, and teams that want to shift towards a model that embraces rapid delivery require an approach with integrated CI/CD. According to a 2017 Forrester Wave report, "CI enables software development teams to work collaboratively, without stepping on each other's toes, by automating builds and source code integration to maintain source code integrity." A continuous delivery approach automates testing and deployment capabilities so that software can be developed and deployed rapidly, reliably, and repeatedly with minimal human intervention.

Waterfall and Agile methodologies aid in organization and prioritization, but they don't help teams thrive in a changing software development landscape.

PLANNING WITH GITLAB

Rather than juggle a variety of tools and struggle with the lack of visibility and communication, teams can use GitLab to work in a single, shared space where teams of all types and sizes can easily and efficiently contribute, get all the information needed in one place and see changes in real time. Regardless of your methodology—from Waterfall or Agile to DevOps—GitLab's simple and flexible approach to planning meets the needs of small teams to large enterprises, keeping everyone synchronized. GitLab enables portfolio planning and management through epics, groups (programs), and milestones to organize and track progress. With self-documenting and self-tracking features, GitLab makes the plan stage one of the most efficient stages of the software development lifecycle.

Connecting teams to accelerate innovation, GitLab comes with everything teams need to plan and track projects and releases, including issues (tracker and board), milestones, burndown charts, epics, and roadmaps. Communication is centralized, plans and progress are visible, and work is linked, making collaboration frictionless. GitLab helps teams organize, plan, align, and track project work to ensure teams are working on the right things at the right time, while maintaining end-to-end visibility and traceability of issues throughout the delivery lifecycle from idea to production.

Everything starts with an issue

GitLab's Issue Board is a simplified approach to a complex problem. Built on top of GitLab's existing issue-tracking functionality and leveraging the power of labels by utilizing them as lists on a Kanban board, teams can construct different views while maintaining the same filtering and sorting abilities seen across the issue tracker to create multiple boards that capture every layer of visibility to define the scope by milestone, labels, assignee, and weight.

An epic next step

An epic is similar to an issue in that it records a proposed scope of work to be done and allows for team members to discuss that scope. However, an epic exists at the group level, so if a feature requires a larger scope and higher level of discussion, epics are used to contain a number of issues. Since an epic is designed to scope work over a longer period of time, a timeline-based view in the form of a roadmap is also useful, because it serves as a visualization to anticipate work and track progress.

The big picture

In comparison to traditional settings, planning with GitLab lets teams organize work with a big picture in mind and allows for information to be viewed in different ways. Team members enter information once (in an issue) and can tag it with a label, a milestone, an assignee, and due dates, creating metadata on top of the work abstraction. The issue can then be organized in boards or lists, connected to related issues, shown in a roadmap view via an associated epic or merge request. So, team members enter information once, but depending on a use case or needs, the data can be consumed in a way that is appropriate for each user.

» A developer is laser-focused on an assigned issue and focuses on the list of requirements within the description. It's nice to know some of the bigger-picture ideas, but a developer is mostly interested in shipping code.
» A product manager needs to view multiple issues together, and within a few clicks she can find an issue board, track progress, and view a burndown chart.
» IT leadership is interested in progress and can use the data to view a roadmap of the next few years. A director doesn't have to ask a question about progress which then trickles down through four people to get an answer.

Everything is in GitLab as data, as a single source of truth, and can be viewed at any time. A single source of truth "orchestrate[s] all parties responsible for release and provide[s] visibility and traceability into who did what and when." In a traditional workflow, this level of access is impossible to get in real time, but it is possible with GitLab.

CONCURRENT DEVOPS

Planning is an integral part of Concurrent DevOps, the new way of thinking about how we create and ship software. Rather than organizing work in a sequence of steps and handoffs, which creates silos, the power of working concurrently is in unleashing collaboration across the organization. In a concurrent world, people have visibility into the entire workflow, process, and security and compliance across the DevOps lifecycle.

Concurrent DevOps makes it possible for Product, Development, QA, Security, and Operations teams to work at the same time. Teams work concurrently and review changes together before pushing to production. Everyone can contribute to a single conversation across every stage, and developers see all information for any change.

Under a Concurrent DevOps model, traditionally disparate teams that work on software development and delivery work together from a single application, sharing a single codebase, datastore, installation, interface, overview, and workflow. Concurrent DevOps gives all stakeholders the real-time visibility and efficiency needed to ship features at the speed the customer demands. For development and operations teams, this model breaks down silos, creating an environment in which teams can develop and operate with confidence.

Benefits of a comprehensive plan stage

When everyone contributes to a single conversation, development and operations can collaborate and contribute to new features early and often in the lifecycle, dramatically shortening cycle times and accelerating delivery.

Benefits to business stakeholders

» Easy measurement of the efficacy of the entire process, characterizing the flows with value streams, providing actionable insight for continuous improvement.

» The ability to organize ideas into transparent, multi-level work breakdown plans that cut across departments.

» Value streams help visualize and manage the flow of new innovation from idea to customers to drive continuous improvement based on data.

Benefits to development and operations teams

» Increased collaboration with team members, using the same set of single-source-of-truth artifacts in a single application.

» Teams can track the execution of plans, adjusting them as needed, to ensure that resources are properly allocated.

» Improved communication between development and operations, since both teams have visibility into progress and discussions.

CONCURRENT DEVOPS WITH GITLAB

Today, only GitLab eliminates the need to manually configure and integrate multiple tools for each project. DevOps teams can start immediately and work concurrently to radically compress time across every stage of the DevOps lifecycle. GitLab is the only vendor offering a single application for the end-to-end application development lifecycle, with features that simplify and accelerate the planning stage.

With GitLab, teams are able to establish visibility into the end-to-end DevOps pipeline so that everyone is aware of pipeline status and can contribute to overall success.

Start your free trial

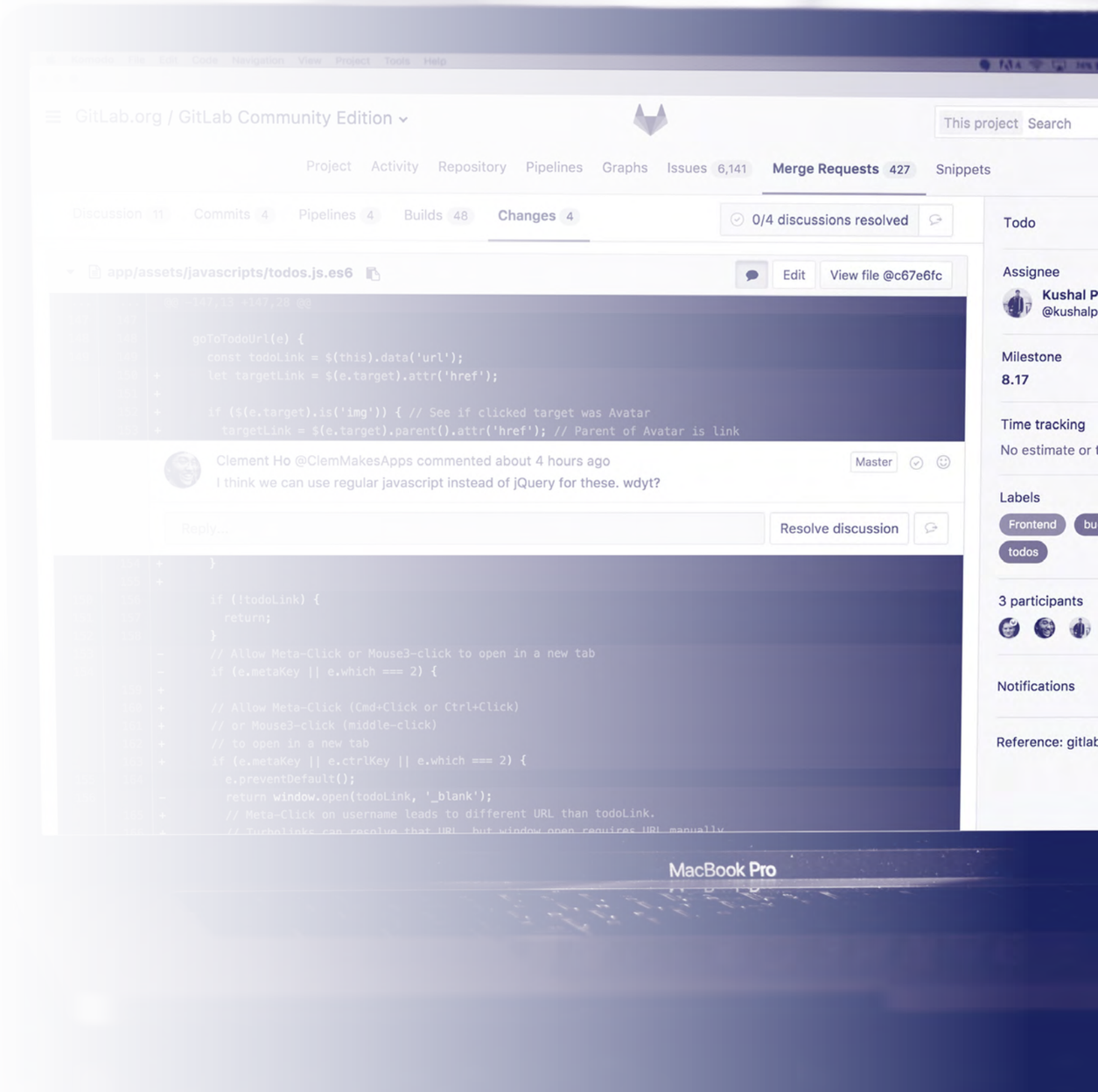
Have questions about GitLab? Contact us!

ABOUT GITLAB

GitLab is the first single application for the entire DevOps lifecycle. Only GitLab enables Concurrent DevOps, unlocking organizations from the constraints of today's toolchain. GitLab provides unmatched visibility, radical new levels of efficiency and comprehensive governance to significantly compress the time between planning a change and monitoring its effect. This makes the software lifecycle 200% faster, radically improving the speed of business.

GitLab and Concurrent DevOps collapses cycle times by driving higher efficiency across all stages of the software development lifecycle. For the first time, Product, Development, QA, Security, and Operations teams can work concurrently in a single application. There's no need to integrate and synchronize tools, or waste time waiting for handoffs. Everyone contributes to a single conversation, instead of managing multiple threads across disparate tools. And only GitLab gives teams complete visibility across the lifecycle with a single, trusted source of data to simplify troubleshooting and drive accountability. All activity is governed by consistent controls, making security and compliance first-class citizens instead of an afterthought.

Built on Open Source, GitLab leverages the community contributions of thousands of developers and millions of users to continuously deliver new DevOps innovations. More than 100,000 organizations, including Ticketmaster, ING, NASDAQ, Alibaba, Sony, and Intel trust GitLab to deliver great software at new speeds.




GitLab