An Agile iteration with GitLab

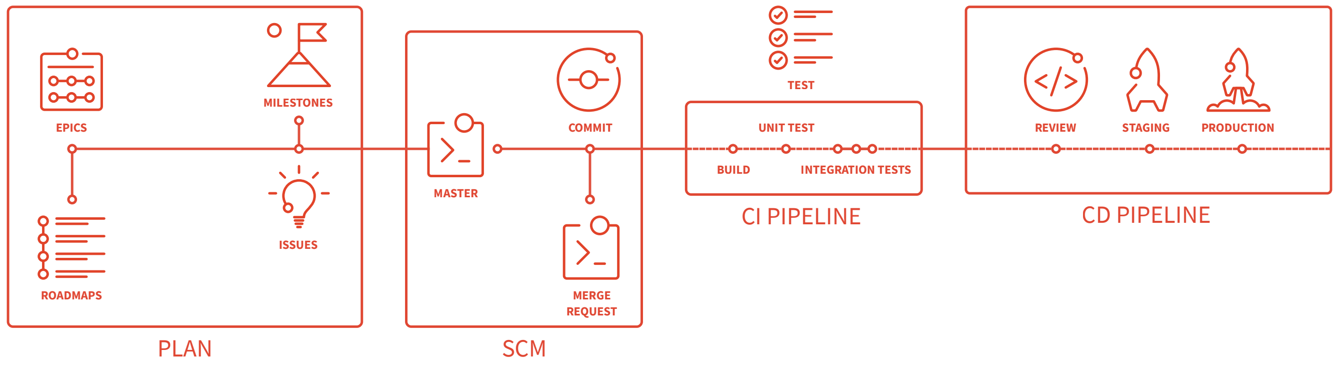
GitLab

What's inside?

Overview

Agile is one of the most important and transformative methodologies introduced to the software engineering discipline in recent decades. While not everyone can agree on the detailed terminology of Agile concepts, it has made a significant positive impact on software teams' ability to efficiently create customer-centric products. The history of Agile and its many transformations are well documented — from the introduction of Waterfall in the 1970s and Scrum and XP in the 1990s to the Agile Manifesto in the early 2000s and SAFe (Scaled Agile Framework) and LeSS (Large Scale Scrum) in more recent times.

Agile is a strong complement to organizations looking to implement DevOps; however, there is significant complexity in creating a toolchain to help organizations accomplish their DevOps goals. This complexity often results in delays in the end-to-end DevOps process, since bottlenecks surface at the integration points of different tools.



GitLab has been designed to be flexible enough to adapt to your methodology and help your team embrace Agile efforts. With GitLab, your efforts can seamlessly tie into your organization's current DevOps initiative. This whitepaper maps Agile artifacts to GitLab features, and explains how customers have successfully run high-performing Agile teams with GitLab.

GitLab for Agile software development

Epics → GitLab epics

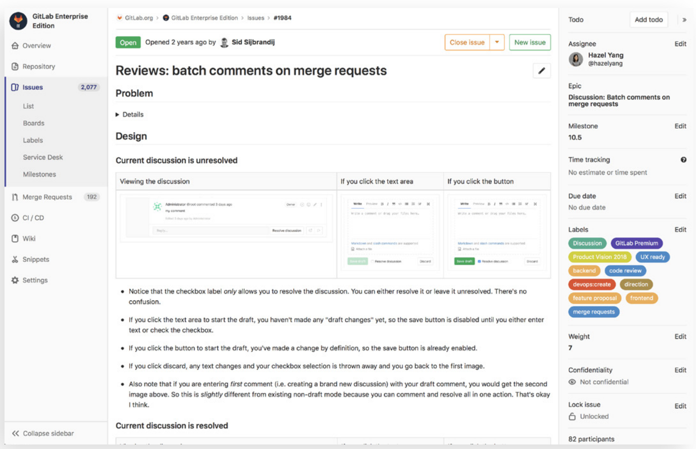
Some Agile practitioners specify an abstraction above user stories (and sometimes "Features"), often called an Epic, that indicates a larger user flow consisting of multiple features. In GitLab, an epic also contains a title and description, much like an issue, but it allows you to attach multiple sub-epics or child epics to it to indicate hierarchy. Epics can span across multiple teams, on multiple projects, and can even be tracked on multiple boards. Epics are almost always delivered over a set of releases and/or sprints.

Roadmaps → GitLab Roadmaps

Roadmaps, a time stamped view of company initiatives, are an accumulation of epics, features, and issues. GitLab Roadmaps give you visibility into Epic progress over a period of weeks, months, or quarters. Epics can be given fixed start and end dates or have a collection of issues roll-up into an epic and the start/end date will be determined by the issues' assigned time box. GitLab Roadmaps provide great visibility into the progress of high level initiatives.

User stories → GitLab Issues

When using an Agile methodology, you often start with a user story to capture a single feature that delivers business value for users. In GitLab, a single issue within a project serves this purpose. Creating issues in GitLab increases transparency, improves collaboration, and allows teams to have a better understanding and focus on customer needs. As you build out your backlog with issues, prioritization helps with coordination of customer needs and helps eliminate technical risks that may affect the business. Issues are an integral part of the different Agile methodologies, including Kanban, Scrum, SAFe, and more. GitLab Issues allow teams to know why they are building and what value it generates.



Open Opened 2 years ago by Sid Sijbrandij

Reviews: batch comments on merge requests

Problem

▸ Details

Design

Current discussion is unresolved

Viewing the discussion	If you click the text area	If you click the button

- Notice that the checkbox label *only* allows you to resolve the discussion. You can either resolve it or leave it unresolved. There's no confusion.
- If you click the text area to start the draft, you haven't made any "draft changes" yet, so the save button is disabled until you either enter text or check the checkbox.
- If you click the button to start the draft, you've made a change by definition, so the save button is already enabled.
- If you click discard, any text changes and your checkbox selection is thrown away and you go back to the first image.
- Also note that if you are entering *first* comment (i.e. creating a brand new discussion) with your draft comment, you would get the second image above. So this is *slightly* different from existing non-draft mode because you can comment and resolve all in one action. That's okay I think.

Current discussion is resolved

Assignee: Hazel Yang @hazelyang

Epic: Discussion: Batch comments on merge requests

Milestone: 10.5

Time tracking: No estimate or time spent

Due date: No due date

Labels: Discussion, GitLab Premium, Product Vision 2018, UX ready, backend, code review, devops:create, direction, feature proposal, frontend, merge requests

Weight: 7

Confidentiality: Not confidential

Lock issue: Unlocked

82 participants

The GitLab Issue page has a title and a description area, providing a space to document any details, such as the business value and relevant personas in a user story. The sidebar provides integration with other Agile-compatible features like the epic that the issue belongs to, the milestone in which the issue is to be worked on, and the weight of the issue, reflecting the estimated effort.

Task → GitLab task lists

Often, a user story is further separated into individual tasks. You can create a task list within an issue's description in GitLab to further identify those individual tasks. Tasks are written by the team, for the team, using the language the team understands. A task is a piece of work that combines with other tasks to complete an Issue (user story). Its purpose is not to have independent deliverable functionality or generate business value, but to aid in the issue's deliverable to which it is directly correlated. For example, seeing functions of an issue organized into tasks that are specific to design, code, test, document, or UX. Tasks enable teams to be more cross-functional and work in parallel specific to their functional expertise.

Product backlog → GitLab Issue lists and prioritized labels

Product or business owners typically create user stories to reflect the needs of the business and customers. The user stories are prioritized in a product backlog to capture urgency and desired order of development. The product owner communicates with stakeholders to determine priorities and refine the backlog.

In GitLab, there are dynamically generated issue lists which users can view to track their backlog. Labels can be created and assigned to individual issues, which then allows you to filter the issue lists by a single label or multiple labels, enabling further flexibility. Priority labels can also be used to order the issues in those lists. Additionally, Scoped Labels allow teams to annotate their issues, merge requests, and epics to achieve custom fields and custom workflow states by leveraging a special label title syntax.

Sprints → GitLab milestones

A sprint represents a finite time period in which work is to be completed, which may be a week, a few weeks, or perhaps a month or more. The product owner and the development team meet to decide work that is in scope for the upcoming sprint. GitLab's milestones feature lets teams give milestones a start date and a due date to capture the time period of the sprint. The team then puts issues into that sprint by assigning them to that particular milestone.

Releases → GitLab milestones

Releases can also be organized in the same manner, which may be a few weeks, a month, or more. The product owner and the development team meet ahead of the upcoming release and decide what issues will be on the sprint backlog. GitLab's milestones feature can capture the time period of the release. In Kanban, this approach is most common as sprints are not part of this methodology.

Points and estimation → GitLab issue weights

During scoping meetings, user stories are communicated, and the level of technical effort is estimated for each in-scope user story. In GitLab, issues have a weight attribute, which teams can use to indicate the estimated effort. User stories can be further broken down to technical deliverables, sometimes documenting technical plans and architecture. In GitLab, this information can be documented in the issue, or in the merge request description, as the merge request is often the place where technical collaboration occurs.

During the sprint (GitLab milestone), development team members pick up user stories to work on, one by one. In GitLab, issues have assignees. So team members can assign themselves to issues to indicate what they're working on.

Pull Requests → GitLab Merge Requests

Merge requests, often known as pull requests, allow teams to visualize and collaborate on the proposed changes to source code that exist as commits on a given Git branch. A Merge Request (MR) is the basis of GitLab as a code collaboration and version control platform. It is as simple as the name implies: A request to merge one branch into another. Merge requests can easily be created from within an issue. Once a merge request is closed, it will automatically close the associated issue.

Agile board → GitLab Issue Boards

Throughout the sprint, issues move through various stages, such as Ready for dev, In dev, In QA, In review, and Done, depending on the workflow in your particular organization. Typically, these are columns in an Agile board. In GitLab, issue boards allow you to define your stages and enable you to move issues through them. The team can configure the board with respect to the milestone (where sprints or releases have been defined), and move issues from the product backlog to the release or sprint backlogs.

In Scrum, the issues would be assigned to the upcoming sprint and product owners, enabling Scrum masters and team members to have visibility into issues and the assigned sprint. During daily standups, the team looks at the board together to see the status of the sprint from a workflow perspective. Boards can also be organized by assignee so that product owners and Scrum masters can view various stages of the workflow.

The GitLab issue list pulls in issues relevant to the particular group or project scope. There are powerful filtering and ordering capabilities that allow you to quickly narrow down that list. For example, you can see a product backlog of prioritized user stories by filtering by prioritized labels. You can also anticipate which user stories will be worked on in a particular sprint by filtering by milestone.

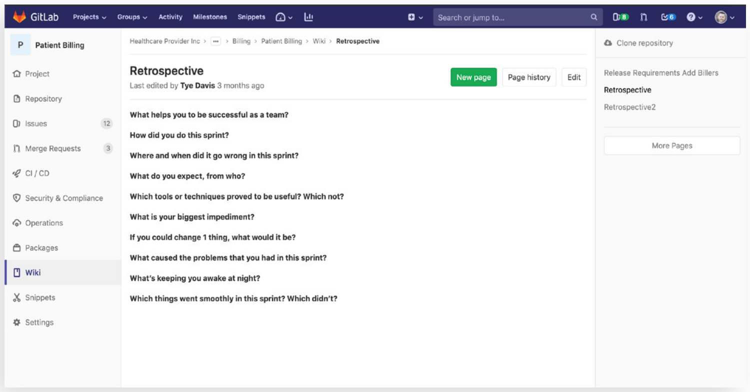
The GitLab Issue Board also pulls in issues dynamically, similar to the GitLab issue list, but it allows for more flexible workflows. You can set up individual lists in the board to reflect Agile board stages. Your team can then control and track user stories . For example, if a story moves from Ready for dev to Released to production, you'll have full visibility into progress.

Burndown charts → GitLab Burndown Charts

The development team wants to know if they are on track in real time and mitigate risks as they arise. GitLab provides burndown charts, allowing the team to visualize the work scoped in the current sprint "burning down" as they are being completed. Teams can react to risks sooner and adapt accordingly. For example, using burndown charts helps teams inform business stakeholders that certain features are anticipated to be delayed to a future sprint.

Toward the end of the sprint, the development team demos completed features to various stakeholders. With GitLab, this process is made simple using Review Apps so that even code not yet released to production, but in various testing, staging, or UAT environments can be demoed.

Review Apps and CI/CD features are integrated with the merge request itself. These same tools are useful for developers and QA roles to maintain software quality, whether through automated testing with CI/CD or manual testing in a Review App environment.


Healthcare Provider Inc › ··· › Billing › Patient Billing › Wiki › Retrospective

Retrospective
Last edited by **Tye Davis** 3 months ago

What helps you to be successful as a team?

How did you do this sprint?

Where and when did it go wrong in this sprint?

What do you expect, from who?

Which tools or techniques proved to be useful? Which not?

What is your biggest impediment?

If you could change 1 thing, what would it be?

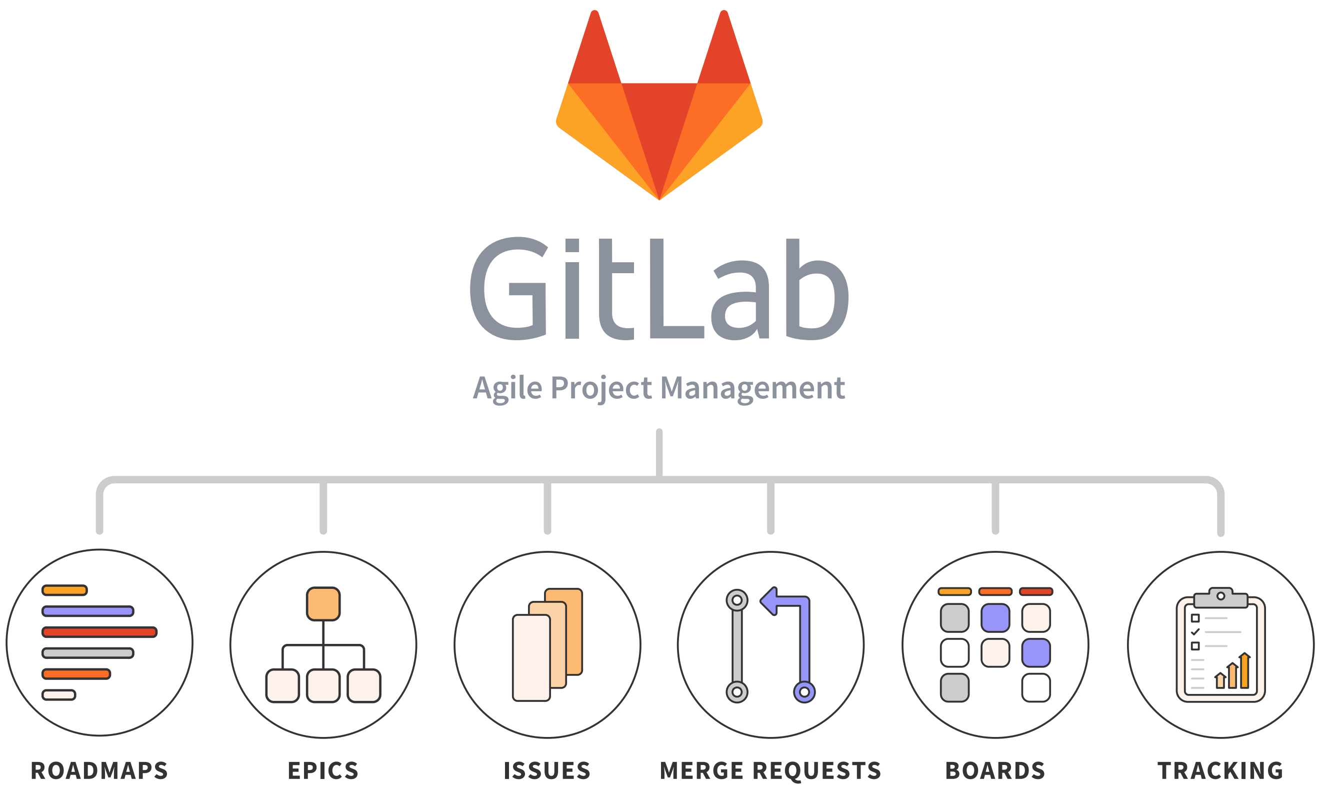
What caused the problems that you had in this sprint?

What's keeping you awake at night?

Which things went smoothly in this sprint? Which didn't?

A team retrospective at the end of the sprint can be documented in the provided wiki, so that lessons learned and action items are tracked over time. During the actual retrospective, the team can look at the milestone page, which displays the burndown chart and other statistics of the completed sprint.

GitLab Agile project management flow



Getting started

High-performing Agile teams use GitLab for Agile project management, starting with Epics and Roadmaps. As organizations take their large ideas and organize them into Epics, the information is further broken down into sub-epics (features) and even smaller into issues (user stories). As teams iteratively complete the issues that have been created from these larger ideas (epics), they contribute to the completion of that epic, with their work displayed in Roadmaps within GitLab. Roadmaps give leadership a high level view of portfolio progress. When issues are created, teams can apply issue weight based on their organization's predetermined scale (e.g. Fibonacci), helping teams select what issues they should work on based on capacity.

Product owners, product managers, Scrum masters, and team members then use Milestones as a way to define their time boxes. With either releases or sprints, GitLab's Milestone feature allows teams to organize based on their agility. GitLab issue boards give teams the ability to groom their product backlogs and organize by either release or sprint boards, depending on their chosen Agile methodologies.

Delivering with ease

Later, developers can create a Merge Request directly from issues and effortlessly checking a branch from master so that the developer can begin work. Merge requests provide visibility into commits, diffs, and pipelines, and once the merge request is closed, it will automatically close the associated issue. Traditionally, developers must manually close a user story with a separate tool, but GitLab automates this step. The direct connection between Agile project management and version control can only be found in GitLab, with no integration needed.

Teams can continually track their progress with their releases and sprints via Burndown Charts located in Milestones. Using Burndown Charts, teams monitor their issues or issue weights for a time boxed period. Following the completion of a release or sprint, teams conduct retrospectives using the Wiki feature to reflect on what worked and what didn't work for that sprint or release.

GitLab for your Agile teams

GitLab has been designed to be flexible enough to adapt to your methodology and help your team embrace Agile efforts. GitLab provides end-to-end DevOps, encapsulating 10 defined stages: Manage, Plan, Create, Verify, Package, Secure, Release, Configure, Monitor, and Defend. Agile project management resides under the Manage and Plan categories and is the starting point in the DevOps lifecycle.

As the only tool that offers one complete application for Agile project management, version control, continuous integration/delivery/deployment, packaging, security, and monitoring, GitLab removes the complexity of DevOps toolchains.

For more information on GitLab's Agile project management features, please view our educational videos on how to setup your organization.

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.

Start your free trial

GitLab