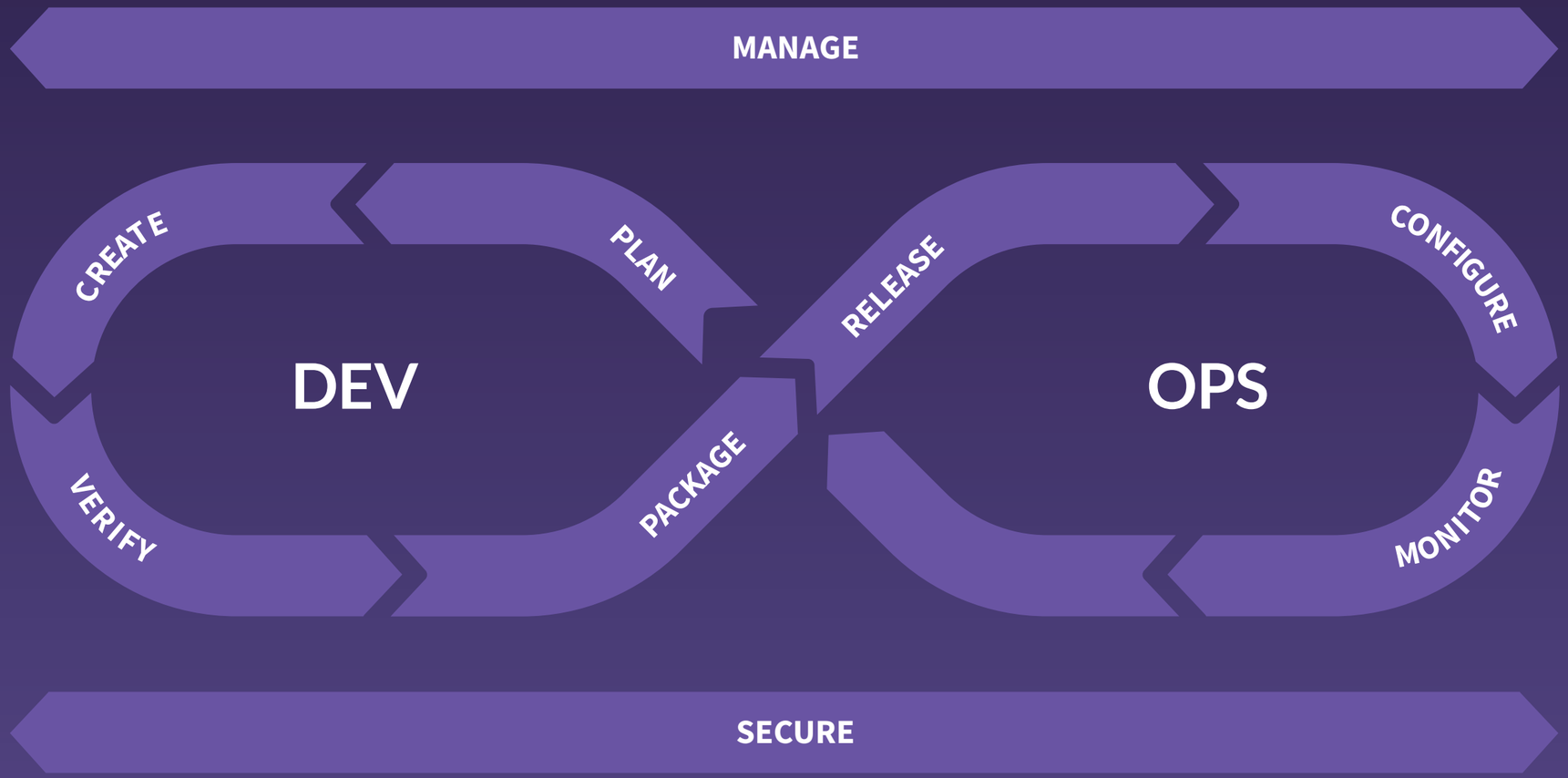
CONCURRENT DEVOPS

HOW TO REDUCE CYCLE TIME, IMPROVE DEVOPS EFFICIENCY, AND SHIP 200% FASTER



GitLab

WHAT'S INSIDE?

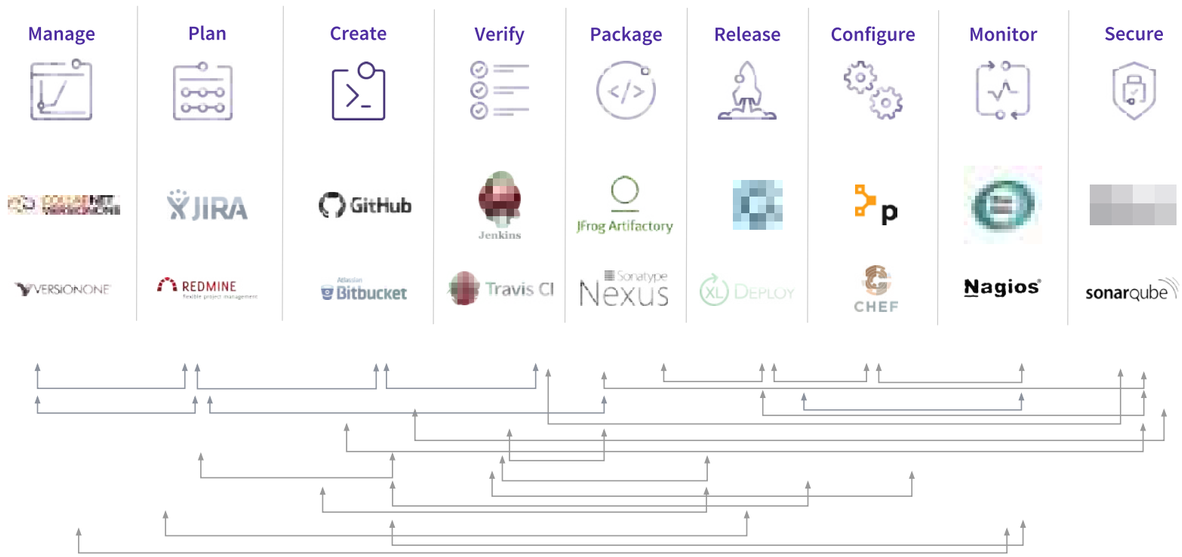
INTRODUCTION

With unprecedented speed and vigor, the digital era of business has arrived. Every executive leader knows software powers their business and has become the new face of the company. Competitive advantage depends on how fast teams can deliver code to provide value to customers. Increasingly, business will be won or lost based upon IT's ability to deliver great software at new speeds.

Business pace is faster than IT pace, and product development teams have been adopted different methodologies to keep up with the business's pace of innovation. Hoping to solve the challenges of a slow and error prone waterfall process, teams adopted Agile and then DevOps. Despite a $3.9B spend on DevOps software last year, more than half of time spent on DevOps is wasted on logistics and repeatable tasks. While some aspects of the lifecycle have been automated, it's not moving the dial. In fact, almost 90% of the organizations that have already adopted DevOps are disappointed with the results, according to a 2017 Gartner report.

While improvements have been made for specific KPIs, like deployment frequency and change failure rate, ultimately, the full transformation promise of DevOps has not yet been realized. Even though teams have implemented DevOps, it's not working how it should. This is because of the very toolchain we're using to try to accelerate the pace of innovation. In fact, today's toolchain limits a faster DevOps lifecycle.

Today's toolchain limits a faster DevOps lifecycle

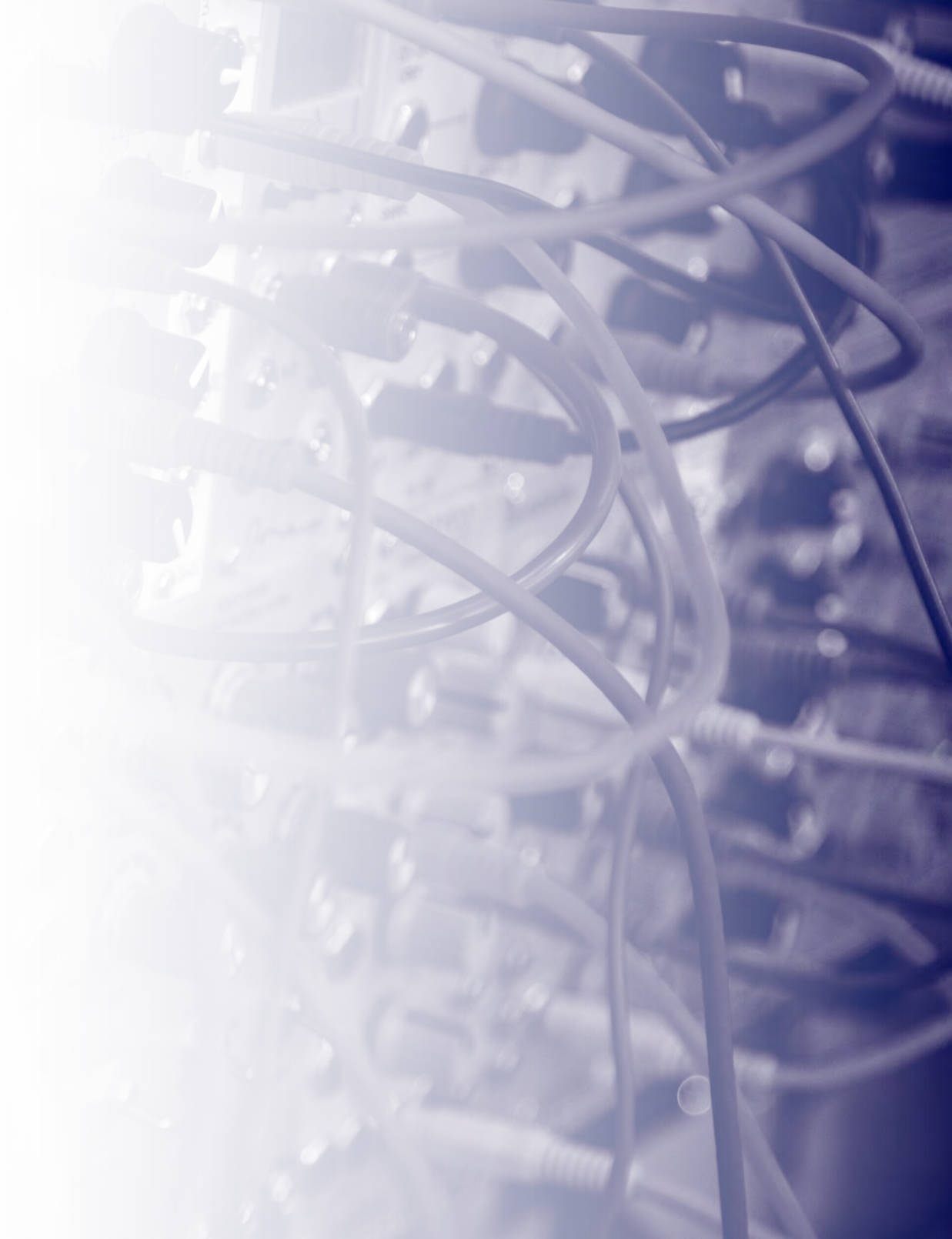


This image shows the core stages of the DevOps lifecycle and the tools commonly used at each stage, illustrating the cost of integration complexity and the reinforcement of silos. Instead of teams collaborating, they're functioning separately due to disparate tools isolating them into different workspaces.

The toolchain and the silos it creates has three main problems:

1. **It's opaque.** Your data lives in multiple data stores across multiple tools, so you can't introspect it. The different silos between teams and their tools makes it difficult to gain an overall visibility of which stage slows down the lifecycle. In order to measure or track progress, additional sets of tooling must be built on top of a toolchain, leading to more integration complexity.
2. **It's inefficient.** Individual silos also mean that teams still rely on manual, semi-automated or automated handoffs. Even when fully automated, one team cannot start their work until the previous teams finishes. This is a form of DevOps, but the workflow is still very much sequential.
3. **It's uncontrolled.** Different tools have different permissions and security policies, preventing control over an environment, making it difficult to establish trust. Furthermore, because security is an afterthought, shipping secure and compliant software is difficult, because it's hard to enforce security in every tool. In fact, your security team probably uses yet another tool to ensure compliance.

The reality is that even though some or all teams may adopt DevOps (i.e. tools, processes, and culture), they often optimize locally but still work sequentially, limiting the organization from reaching the maximum value of their DevOps transformation. By removing the handoffs of work from one team to another, teams can better work together, following a single conversation around smaller deliverables. Cross-functional teams need to be enabled to work concurrently with autonomy and self-service.



BRIDGING THE SILOS

Traditional DevOps typically focuses on the intersection between Dev and Ops, looking at how developers get their code into production and how operations can monitor and manage applications and infrastructure. In an ideal DevOps world, this definition expands to include security and business professionals. Each group gets an experience tailored to their needs, but shares the same data and interface as everyone else so collaboration is easy.

Imagine: an Ops person finds an issue in production, drills down to find the application with the problem, and sees that a recent deploy caused the problem. Simultaneously, a developer is alerted that their recent deploy triggered a change in production, and goes to the merge request to see the performance change right there. When Dev, Ops, and Security talk, they're looking at the same data, but from their own point of view.

Advantages of a single application for the entire DevOps lifecycle

Disparate tools can cause siloing of people and information into different systems. When DevOps teams can work from a single application, the result is happier teams, improved cycle times, and a great return on investment.

1. **Single interface.** A single interface for all tools means that you always have access to all relevant context. You don't lose information due to constant context switching between different interfaces.
2. **Single data-store.** With a single data-store, you get a single source of truth across the whole software development lifecycle. Multiple data-stores lead to redundant and inconsistent data.
3. **Single overview.** Automated links between environments, code, issues, and epics provide a better overview of project state and progress. All information is real-time and there is a coherent set of concepts and definitions.
4. **Single flow.** Using a single application means you don't have to integrate 10 different products. This saves time from your developers, reduces risk, increases reliability, and lowers the bill of external integrators.

CONCURRENT DEVOPS

Concurrent DevOps is a 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, 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 relevant security and ops 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. This removes the need for messy integrations, requesting access to each tool, and context switching. All the tools needed, from issue tracking and project management to code review, continuous integration, and release automation are all inherently connected into a single, uninterrupted, and concurrent workflow.

Traditional DevOps workflow vs. Concurrent Devops workflow:

Microsoft Word	Google Docs
One person edits at a time	Many people edit at the same time
Multiple copies (hand offs)	One copy
Version conflicts	No conflicts
Waiting for feedback	Realtime feedback
Sequential	Concurrent

Traditional software development is typically sequential, not unlike how documents are developed using Microsoft Word. Concurrent DevOps is different; teams can collaborate and contribute to new features early and often in the lifecycle, dramatically shortening cycle times and accelerating delivery.

THREE PILLARS OF CONCURRENT DEVOPS

Visibility

Concurrent DevOps provides complete real-time visibility of all projects and relevant activities across the expanded DevOps lifecycle. Stakeholders should be able to easily see changes, cycle times, security, and operational health. Information is shown where it matters most:

1. Production impact is shown together with the code change that caused it.
2. Developers see all relevant security and ops information for any change in their merge request.
3. A trusted, single source of truth stays up-to-date with information across the lifecycle without the need for a monitoring app to sync to version control.

This level of visibility eliminates the need to copy issue status or manually link separate projects across tools by providing progress status and cycle time patterns across all project activities in one place.

Efficiency

The Concurrent DevOps model eliminates the need to manually configure and integrate multiple tools for each project. As soon as you push your first code change, the language of your code should be detected, your project packaged, security tests run, code quality reviewed, and deployed with a review app automatically.

This helps developers detect problems earlier by "shifting them to the left" and solving them without delays. If there's an issue in the process, Development, QA, Security, and Operations teams can all contribute to a single conversation throughout the lifecycle. Because all of the information is housed together, teams can work at the same time instead of waiting for handoffs. Security is tested for every project as soon as it is pushed. The security state of all projects is visible across the company. Security, QA, and Operations are no longer a roadblock preventing a faster DevOps lifecycle.

Governance

Embedded automated security, code quality, vulnerability management, and policy enforcement keep things moving quickly while remaining compliant. A single permission model where user authentication and authorization is enforceable and consistent removes the need to manage multiple authorization schemas across several applications and enables consistent user rights across all product categories.

With the Concurrent DevOps model, every important activity is logged in a single audit log that covers the entire DevOps lifecycle—always on, accessible, and accurate—to allow tight control over how code is deployed, eliminating guesswork.

BENEFITS OF CONCURRENT DEVOPS

Concurrent DevOps allows businesses to realize the full potential of their digital transformation. Giving cross-functional teams the ability to work concurrently, instead of sequentially, has the potential to improve the speed of the DevOps lifecycle by 200%.

Benefits to the business

» Faster financial results improves time to value.

» Fewer projects in flight reduces the number of cancelled projects.

» Rapid iteration based on feedback improves customer experience.

Benefits to the IT organization

» Small steps reduce coordination overhead.

» More interactions increase information and creates predictable progress.

» Increased decision points prevent all-or-nothing situations, creating more project control.

Real life stories

The benefits of Concurrent DevOps are already being realized by companies across the globe.

Paessler AG

Ramped up to 4x more releases by eliminating cumbersome, sequential processes.
Read their story.

“Every branch gets tested, it’s built into the pipeline. As soon as you commit your code, the whole process kicks off to get it tested. The amount of effort involved in actually getting to the newest version that you’re supposed to be testing, whether you’re a developer or a QA engineer, is minimized immensely.”
— Greg Campion, Sr. Systems Engineer

Axway

Realizes a 26x faster DevOps lifecycle with more intelligent automation and developer self-service. Read their story.

“Our legacy toolchain was complex, disconnected, and difficult to manage and maintain. Now, developers are empowered to self-serve with minimal guidelines.”
— Eric Labourdette, Head of Global R&D Engineering Services

Reduced cycle time

A shorter cycle means it takes less time to go from creating an idea to implementation monitoring, and scaling.

By shortening this time, you're able to faster respond to changing needs from the market, helping you to adjust long-term plans based on feedback, and radically reduce engineering risk with small batch sizes.

Shorter feedback cycle

A shorter cycle time allows you to immediately respond to changing needs. Rather than having to wait upwards of several months for the DevOps cycle to complete, with a short cycle you're able to adjust your plans quickly. With each iteration you complete, you get new opportunities to collect feedback. Be that low-level feedback on the performance of your products or direct feedback from your customers.

Reduced engineering risk

Smaller and more frequent deploys reduce engineering risk by making it easier to coordinate and providing a higher predictability of what will ship, when. Smaller batch sizes result in better code quality, because every small change gets the attention it deserves, rather than trying to comb through a mound of changes all at once. This also results in easier troubleshooting: a smaller deploy introduces fewer changes that can potentially introduce issues.

GET STARTED WITH CONCURRENT DEVOPS

To get started with Concurrent DevOps:

1. **Create embedded, cross-functional teams.** At GitLab, we adopted a stable counterparts model to facilitate cross-functional connections in the hope that working with the same people would increase the speed of communication, build trust, and encourage iteration. In a stable counterparts model, every team works with the same team members, including frontend engineers, UX designers, and test automation engineers, for each release, creating a smaller team within GitLab.
2. **Ship minimal viable changes.** Reduce every change proposal to its absolutely minimally viable form. This allows you to ship almost anything within a single release, get immediate feedback, and avoid deep investments in ideas that might not work.
3. **Focus on velocity over predictability.** Optimize for shipping a high volume of user/customer value with each release. Eschew heavyweight processes like detailed story point estimation by the whole team in favor of lightweight measurements of throughput like the number of merge requests that were included or rough estimates by single team members.

Concurrent DevOps with GitLab

Today, only GitLab eliminates the need to manually configure and integrate multiple tools for each project by providing everything DevOps teams need in a single application. Teams can start immediately and work concurrently to radically compress time across every stage of the DevOps lifecycle. With GitLab, there is never any need to wait on synchronizing your monitoring app to version control or copying information from tool to tool.

GitLab frees teams to manage projects, not tools. These powerful capabilities eliminate guesswork, help teams drive accountability and gives everyone the data-driven confidence to act with new certainty. With Gitlab, DevOps teams get better every day by having the visibility to see progress and operate with a deeper understanding of cycle times across projects and activities.

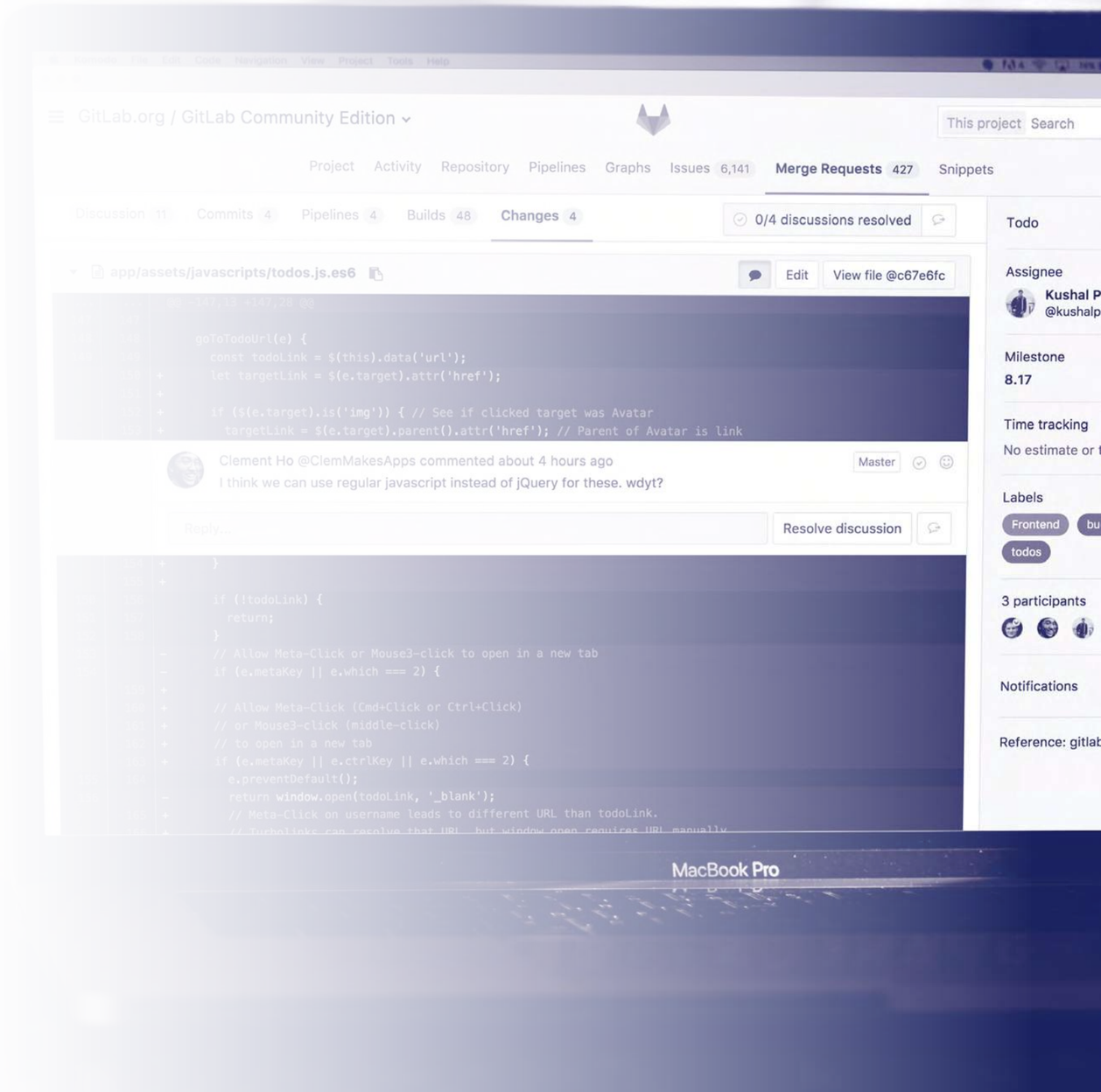
Start Your Free Trial

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. With more than 100,000 organizations from startups to global enterprise organizations, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network and Comcast trust GitLab to deliver great software at new speeds.




GitLab