A SEISMIC SHIFT IN APPLICATION SECURITY

HOW TO INTEGRATE AND AUTOMATE SECURITY IN THE DEVOPS LIFECYCLE

WHAT'S INSIDE?

INTRODUCTION

Application Security Testing has been around for a long time, yet applications continue to be a lucrative target for attackers. CIO magazine recently summarized the top attacks in, "What is a cyber attack?" Recent examples show disturbing trends." Four of the top six attacks were application based. One of the the most infamous, WannaCry, exploited a vulnerability in Microsoft Windows using code that had been secretly developed by the United States National Security Agency. Microsoft had already patched the vulnerability a few weeks before, but many customers had not updated their systems. Similarly, the Equifax attack was targeted at their website application by exploiting a known vulnerability in common third-party code (Apache Struts2). The flaw made it possible for the attacker to send malicious commands that enabled access to files with sensitive data.

These attacks happen despite significant investments in application security. It's logical to conclude that current approaches are outdated and inefficient. Fortify, an early leader in application security testing, was founded in 2002, sixteen years ago, well before agile and DevOps flourished and when cloud computing was a novelty. Even relative newcomers such as Contrast Security and Checkmarx do not go far enough to change the inefficient application security workflow—they can't without being tightly integrated into the development workflow itself. Today, that means stitching together an elaborate patchwork of DevOps and application security tools, each focused on a narrow element of the software development lifecycle (SDLC). Everyone claims to "shift left," but how far left? And is that enough? Automation and Concurrent DevOps have made Agile development capable of lightning speed and better quality code at the same time. Can we apply the same principles to application security?

Security can be expensive, and thus an enterprise may often choose to scan only critical applications, or test applications infrequently, allowing introduced vulnerabilities to go unaddressed (unpatched) for some time. Security requires costly and scarce security professionals to run the test, interpret the findings, and triage the results. Application security is least understood, and often takes a back seat to perimeter and endpoint security.

Likewise, there is a misconception with cloud computing that the cloud provider takes care of ALL the security. While cloud providers have indeed taken on securing most layers of the OSI infrastructure stack, they lack insight inside the application, and the context needed to identify and protect application security vulnerabilities. As a result of these challenges and misconceptions, application security testing is largely neglected, and when there is coverage, it's riddled with holes.

THE TRADITIONAL APPLICATION SECURITY WORKFLOW

Traditional application security testing has been targeted to security professionals and is regarded as a separate process from development. With this approach, security is a step at the end of the development lifecycle where a security professional runs tests to identify vulnerabilities, vets each vulnerability for false positives, prioritizes them by risk, and triages for remediation. With security acting as a gate for code deployment, it's sometimes perceived as a roadblock to innovation by developers and engineers. The intent is to test all of the code but often the volume is too high, the tests may take too long to run, and there's a shortage of security analysts to triage, prioritize, and validate results.

Because the effort is siloed from the development workflow, it may be some time before the developer learns of the needed remediation. By then, they have moved on to another project and must take some time to re-engage. This separation and delay creates friction in the process, with many trade-offs required.

It's important to strike a balance between managing risk and business agility. You know the low-risk approach would be to test everything, all of the time. But that's not practical given the high cost of application security testing tools or services and the scarcity of security professionals. Nor is it prudent to minimize testing to only that which is required for regulatory compliance—Target was compliant and passed their audit but was still famously hacked. Attackers look for the weakest link, like internal applications and unpatched software.

Trade-offs in a traditional Application Security model

You want to test everything but you...	So you may...	Even though...
Can't afford to test all of your code or have everyone use scanning tools.	Only test externally-facing apps because they have higher risk or only test major releases.	Hackers can traverse laterally across your enterprise. They look for the weakest link.
Lack security analysts to validate results, prioritize and triage.	Rely on infrequent external penetration testing that meets minimum compliance requirements.	Exploits change all the time. Penetration testing only tests a snapshot in time.
Have already found more vulnerabilities than you have time to fix.	Focus on one testing method alone.	Different types of scans are best at finding different types of vulnerabilities.

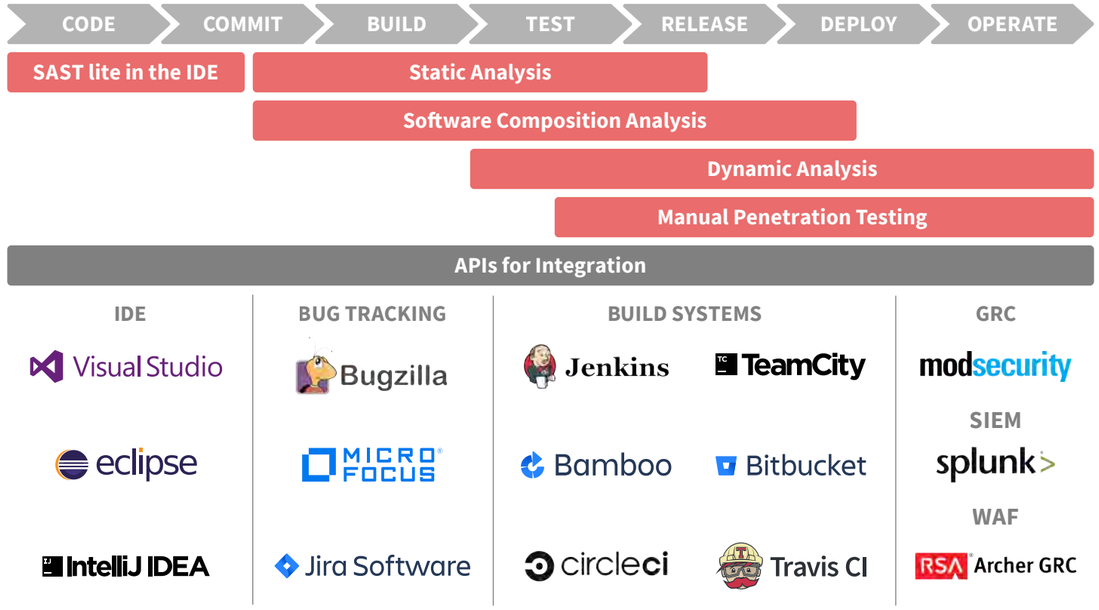
In an effort to improve application security testing, the new chant has been “shift left” to remove more vulnerabilities earlier. The idea is to empower the developers to find and fix vulnerabilities earlier in the software development lifecycle, when changes are less costly and more timely. But shifting left must go beyond simply providing a lightweight static analysis in the development environment (IDE). A spell-check-like functionality is a great idea to help developers find vulnerabilities, and it can help educate them on how to avoid these same mistakes again and again. But it’s just a tiny step in the right direction.

WHY “SHIFT LEFT” ISN’T ENOUGH

Several vendors offer “lite” static testing with a plug-in to the developer’s IDE. The intent is to show developers vulnerable coding practices while they code, so they can learn how to avoid these mistakes, and remediate them immediately, without leaving their native environment. This is a start, yet shortcomings include:

» The cost, maintenance, integration, and user interface for yet another security product.
» Plug-ins are available for only a few IDEs.
» Security results are not always intuitive to developers.
» Vulnerabilities can be easily dismissed with no automated capture, only to resurface later (to be dealt with yet again) in subsequent testing.

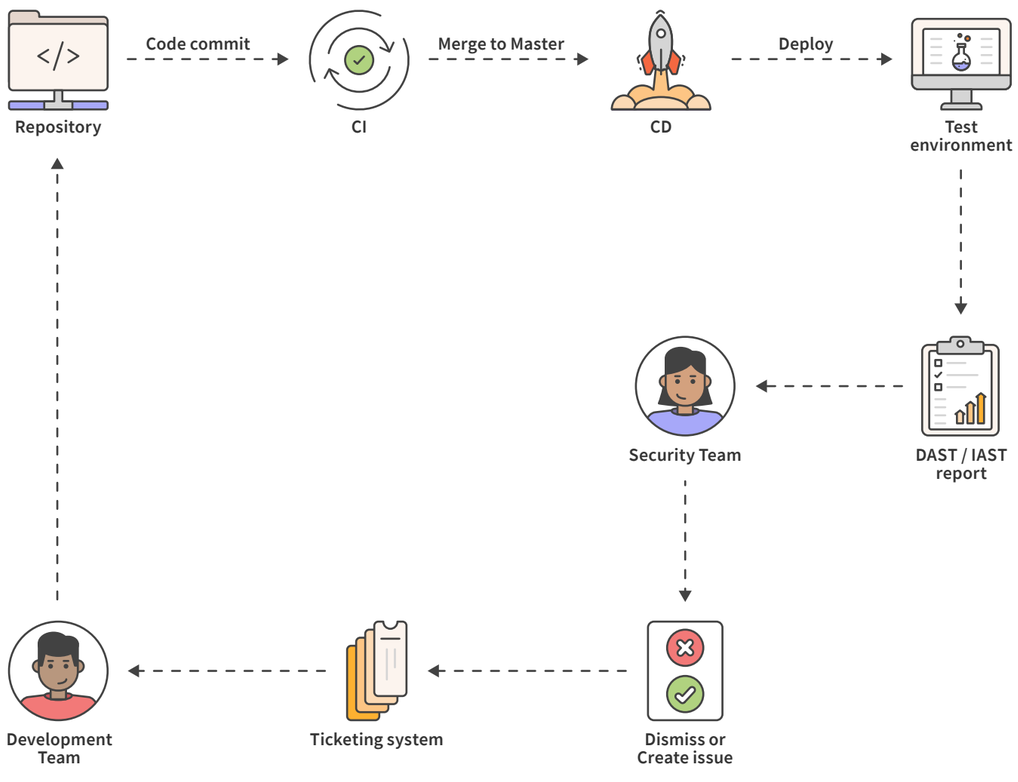
Application Security silos require a web of integrations



Application Security silos require a web of integrations.

Using this approach, the bulk of application security remains a separate workflow. Full static scans and dynamic scans are not possible in the IDE; independent security tools are simply unable to perform dynamic testing on code before it is merged. Interactive application security testing (IAST) is intended to bridge that gap, with dynamic security scans triggered by your already-planned user and integration testing. While this occurs further into the SDLC, once there is a working application, it is still only an incremental improvement.

Traditional DAST without GitLab



Traditional security testing creates process friction. Security runs the scans but must convey findings to developers to remediate.

THE COMING SEISMIC SHIFT

What's needed is a seismic shift left. Traditional application security tools cannot deliver this because they were built for traditional development methods and struggle to stay relevant even in today's Agile environment. DevOps compounds that challenge. Traditional tools struggle with cloud native environments. Scanning containers is often not covered by traditional application security testing tools (leading to yet another tool purchase and integration requirement!). In addition, with iterative development cycles where code may be pushed to production, weekly, daily, or hourly, a security gate at the end of the SDLC, or even in a test environment, just doesn't work. So how do you change that? A seismic shift is needed.

INTEGRATED AND AUTOMATED END-TO-END SECURITY

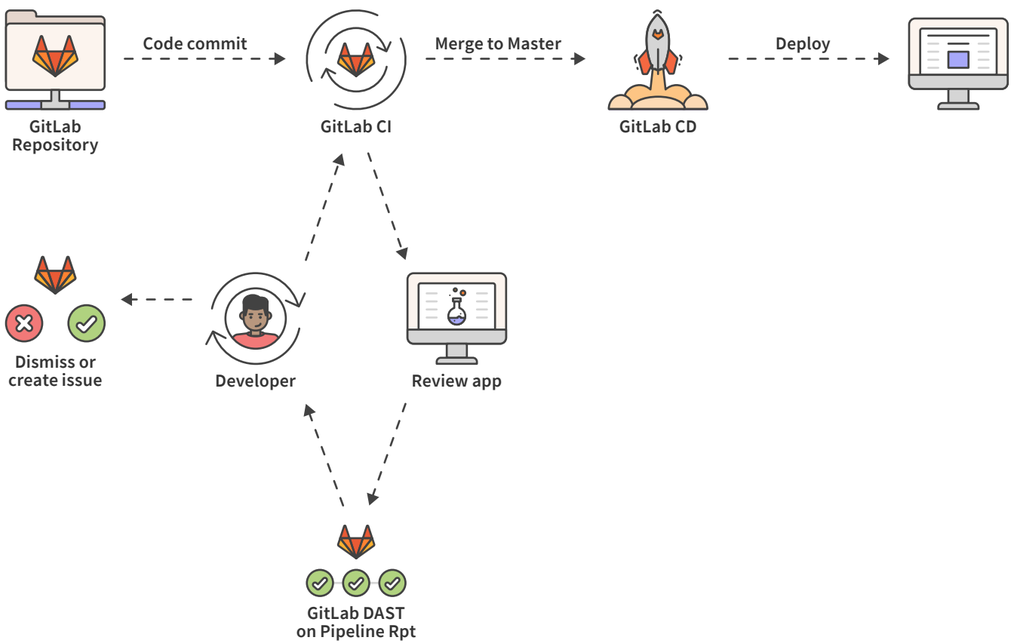
A seismic shift left can only occur by integrating so tightly into the SDLC that code changes are automatically tested with every code commit and results are presented to the developer for instant remediation. The workflow must be a single one, not a separate silo where half of the application security time and budget is spent tracking vulnerabilities for their severity, potential risk to the organization, and time to remediation. By integrating with the CI/CD process, the tracking becomes a by-product of development steps already being taken, and the focus can be on the remediation itself.

Development teams must be seen as the chief users and buyers of application security tools, with security as a co-funder. Even if developers' performance is not measured by delivering secure code, they should be tasked with speed and time to market. If security is a bottleneck to deployment, then the application development leadership will certainly care about security, even if the individual engineer does not.

Checkmarx has been on a path to woo developers. They are integrating into the SDLC via APIs and focusing more on what developers need as users. Their SAST can perform incremental scanning to speed test cycles by testing only the code that has changed; their IAST approach is meant to bring DAST earlier into the cycle and make it more iterative. However, even Checkmarx is limited in their ability to help enterprises truly shift left, because they remain an independent security silo. Their SDLC integration requires a patchwork of APIs and plug-ins. To automate the scan, a DevOps tool like Jenkins must be configured. Although their incremental scanning can speed up testing, it requires use of a sophisticated hashing routine to determine what code has changed.

Alternatively, when the application security tool and the source control tool are the same, there is no need to identify incremental new code by looking from the outside: the code repo can show definitively what has changed. The developer's actions of committing code to the repository kicks off security scans, then the CI/CD component spins up a review app for the developer to test their code interactively before it is merged. Dynamic scans are performed by the developer on the piece of code they are changing, before it is merged with other code. Security is automated and embedded into a development tool they already know (and love!).

GitLab enables interactive DAST testing before the merge



The CI/CD review app enables dynamic scanning to happen before the code leaves the individual developer's workspace.

To achieve a seismic shift left, application security must be part of the development workflow and integrated into the entire SDLC, from source control to CI/CD, and even application monitoring.

THE NEW SECURITY PARADIGM: CONTINUOUS SECURITY TESTING

With security embedded into the development workflow, developers can get feedback on the security of their code as they are working, they can remediate in real time, and free up the security team's time to focus on monitoring issues, assessing risk, and solving vulnerabilities that can't be fixed by the developer. By continuously testing even small, incremental code changes, an avalanche of work is avoided at the end of the SDLC.

Keeping it simple is key. Identifying a false positive can be very subjective, and risk assessment is mostly a human process. Security features should not automatically block a pipeline or prevent a new version from being released to production.

Reports are interactive, actionable, and iterative. When triaging vulnerabilities, users can confirm them (creating an issue to solve the problem), or dismiss them (in the case of false positives). This information will be collected to improve the signal-to-noise ratio that the tool could provide in future executions.

Reports need to be easy to use, and require the minimum amount of effort from users. Otherwise, security checks will likely be disabled or not considered at all, missing their primary goal. Unlike traditional application security tools primarily intended for use by security pros, GitLab's security capabilities are built into the CI/

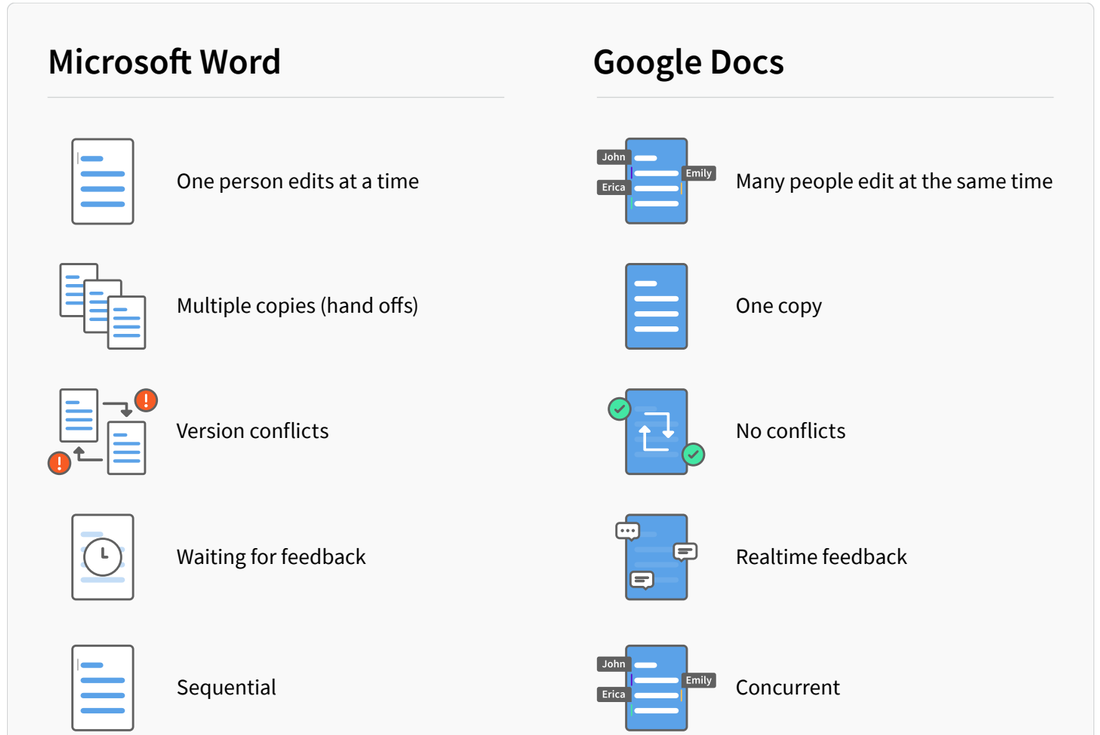
CD workflow where developers live. We empower them to identify vulnerabilities and remove them early, while at the same time providing security pros with a dashboard to view items not already resolved by the developer, across projects. This vulnerability-centric approach helps each role deal with items that are most important and most relevant to their scope of work.

CONCURRENT DEVOPS

Application security 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, working concurrently unleashes 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 relevant security and ops information for any change.

Traditional DevOps workflow vs. Concurrent Devops workflow:



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.

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, efficiency, and governance needed to ship secure code at the speed the business demands. For security

professionals, this model embeds automated security, code quality, vulnerability management, and policy enforcement across the SDLC to keep things moving quickly while remaining compliant. 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 AN INTEGRATED APPROACH

Balancing business velocity with security is possible when application security testing is built into the DevOps workflow. Every merge request is scanned for vulnerabilities in your code and that of its dependencies. Under the Concurrent DevOps model, organizations can develop and operate with confidence, knowing that security and compliance are built in.

Benefits to the business

» Every piece of code is tested upon commit, without incremental cost.

» One source of truth for both development and security promotes collaboration and empowers the developer to identify vulnerabilities within their own workflow.

» Vulnerabilities can be efficiently captured as a by-product of software development.

» A single tool also reduces cost of buying, integrating, and maintaining point solutions.

Benefits to the security organization

» Developers can remediate in real time so security professionals can focus on the vulnerabilities that cannot be easily fixed by the developer.

» An integrated security dashboard displays a roll-up of unresolved vulnerabilities and metrics overall.

» Security can drill-down from the dashboard to see the code itself, and make comments, simplifying collaboration with the developer.

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

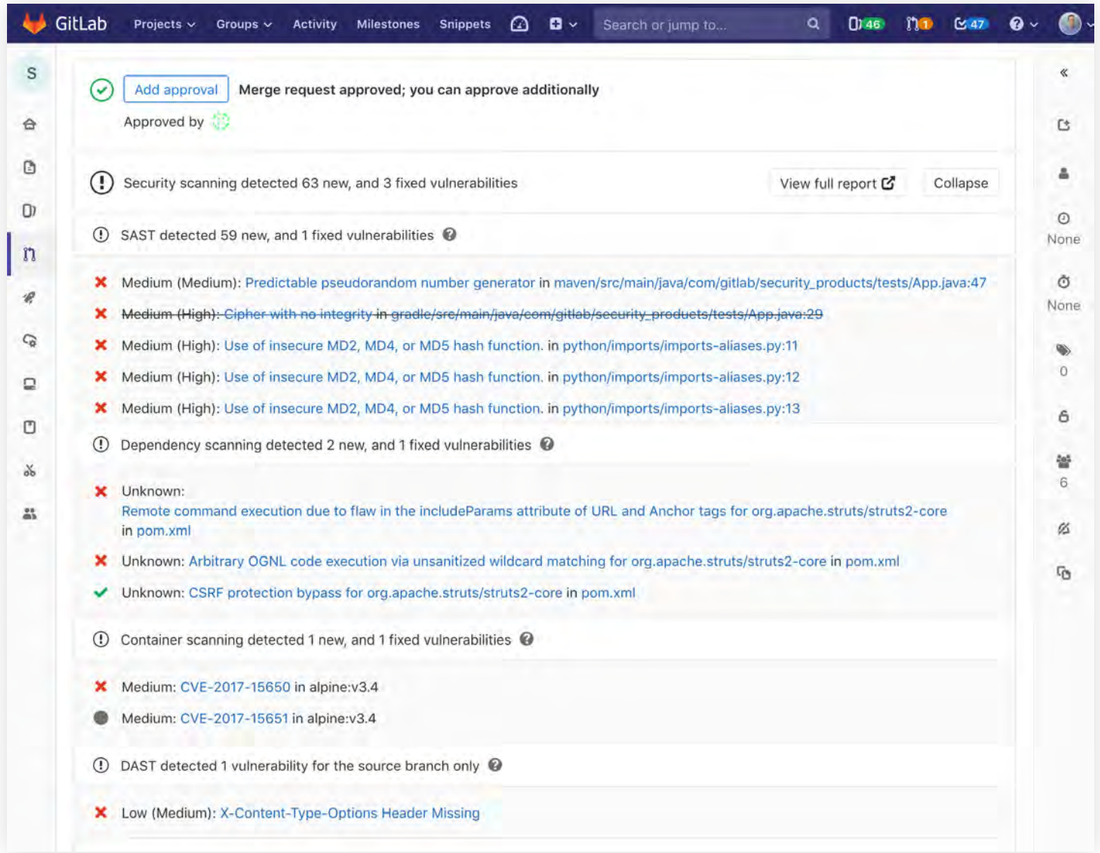
CONCURRENT DEVOPS WITH GITLAB

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, including automated application security testing in the pipeline.

Continuous security testing within CI/CD

Using GitLab, every merge request is automatically tested using static application security testing (SAST), dynamic application security testing (DAST), dependency scanning, container scanning and license management. GitLab's built-in security tests scan the application source code and binaries to spot potential vulnerabilities - without added tools. Developers can remediate from the merge request, and security professionals can evaluate vulnerabilities in context, dismissing or creating an issue in one click.

Similarly, GitLab's DAST capabilities allow for dynamic scanning earlier than previously possible by leveraging the review app generated by the built-in CI/CD. This allows known runtime vulnerabilities to be reported to the developer in-line with every merge request.



Sample security findings within the GitLab CI/CD pipeline.

Start Your Free Trial

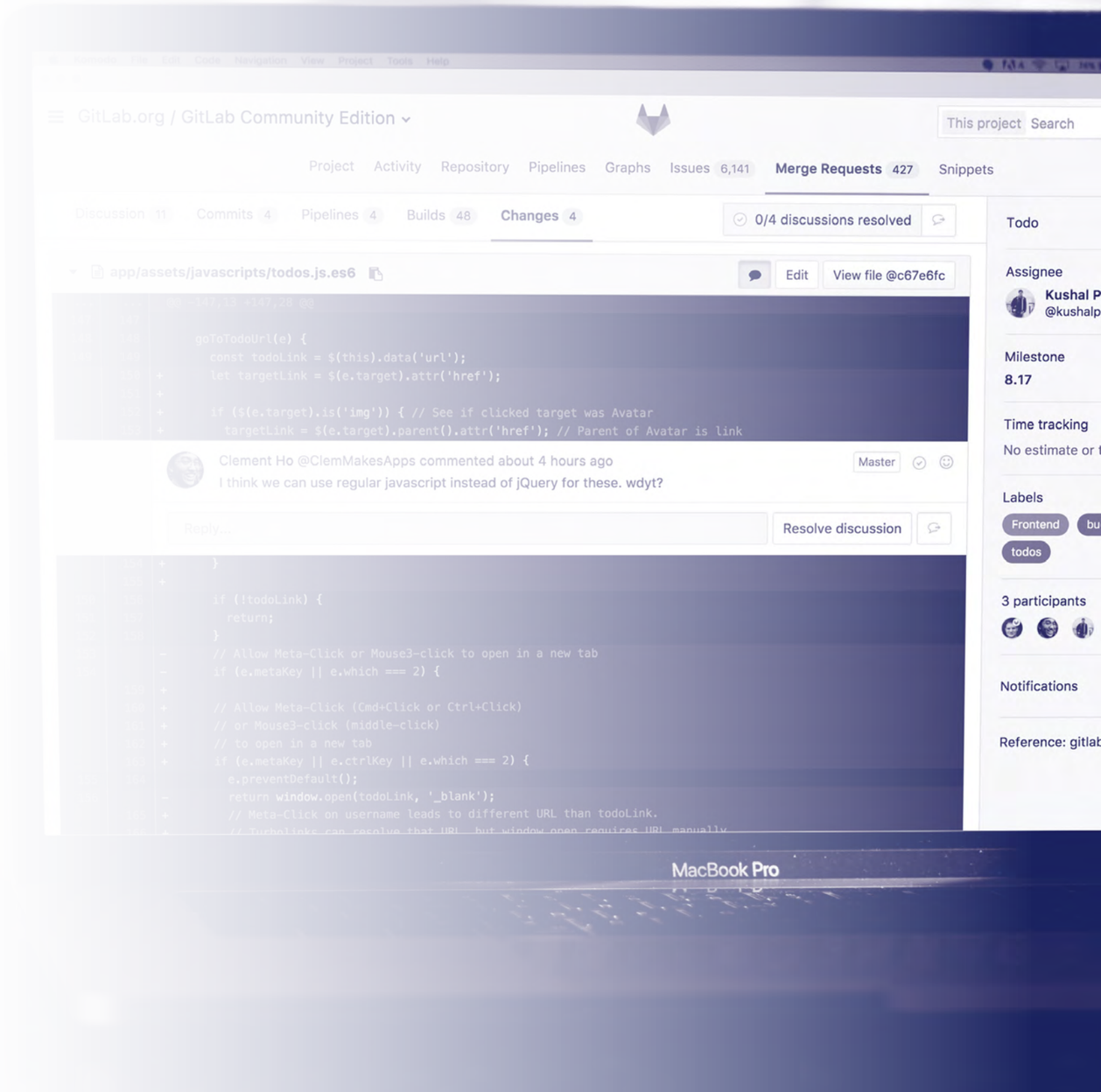
Have questions about GitLab? Contact us!

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. 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