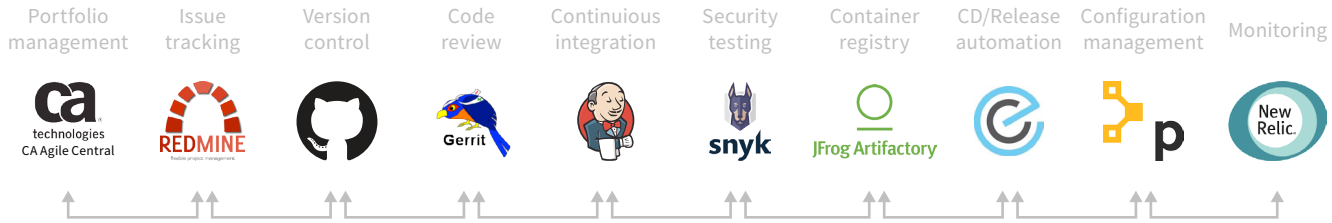
Speed to mission: the strategic value of velocity

GitLab

In today's modern landscape of rapidly changing priorities, new threats, and an increasingly high expectation from citizens for digital services, government agencies — at all levels — must discover new ways to streamline their delivery of digital applications and systems. The traditional approach of defining exhaustive lists of detailed requirements and then waiting years for a contractor or system integrator to deliver a solution is simply no longer a viable option. The pace of change on the battlefield, in the economy, and in our neighborhoods has reached a point where the need for velocity and rapid response is the new imperative.

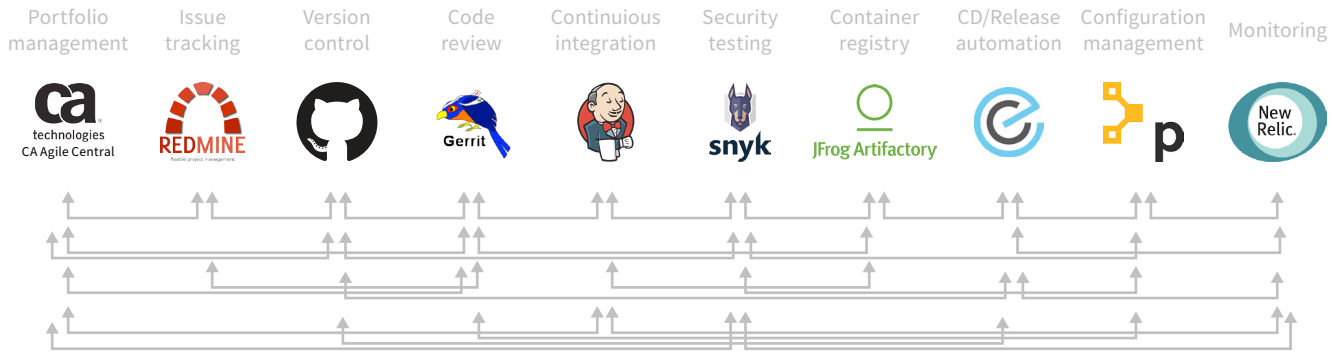
The proven techniques of Agile planning and DevSecOps delivery have demonstrated robust, reliable, and scalable solutions to streamline and accelerate application delivery. New tools to enable automated integration, testing, manage digital assets, scan for vulnerabilities, and repeatedly deploy and configure applications, have given development teams the ability to assemble chains of interdependent tools and tasks required to build, test, and deliver a working application.

A toolchain



While these integrated toolchains help accelerate application delivery, they also introduce new cost and overhead in the form of complexity, islands of data, inconsistent security settings, reporting challenges, and compliance issues. Each new tool adds a new integration and factor of complexity in the entire application delivery team's work. Project managers, developers, testers, operations, and security teams each bring their own tool, contributing to the overall complexity.

A toolchain integration headache



What development teams effectively have is a complex, fragile, expensive Frankenstein tool chain, where they are forced to waste cycle time tinkering on the assembly line tools rather than delivering value. The overhead of managing this DevOps toolchain is like a 'tool chain tax'. What development teams need is a clean and modern software factory with a fully functional assembly line that is efficient, easy to manage, and able to quickly build, test, and deliver their application without the waste and overhead of managing dozens of disparate tools and bespoke integrations.

The software factory must address the following challenges:

1. Issues and planning

Delivery teams must be able to capture, discuss, prioritize, and define new requirements and use cases. New issues serve as the use cases and requirements from end users about the specific capabilities they need. Product managers, delivery teams, and customers define use cases, elaborate on needs, and prioritize new features according to importance. Issues and use cases are effectively the order entry process for the factory and initiate future development.

2. Distributed source code management

Designing and developing applications is an intensive activity that requires managing branches in the source code, tracking frequent changes of multiple files, securing those changes from vulnerabilities, and merging and integrating changes to the main trunk. Distributed source control allows developers to work independently from a centralized repository and keep their work in synch with the larger team. A distributed source code management enables coordination, sharing, and collaboration across the entire software development team.

3. Code reviews and approvals

Peer reviews and rigorous approvals are essential to ensuring that new code changes address user needs and do not introduce logic errors, defects, or security vulnerabilities. Typically, approvals for code changes must be clearly documented and tracked to demonstrate compliance. This critical oversight and review process should be a core capability in the software factory to ensure both quality, accountability, and compliance.

4. Continuous integration for every commit

The backbone of the software factory is the continuous integration pipeline which automates development tasks to be completed for every code change. The CI pipeline ensures the right sequence of automated tests, scans, and compliance checks are completed.

a. **Software quality (automated testing)**

The CI pipeline manages automated testing for every commit, ranging from unit, API, functional and nonfunctional tests. The goal of automated testing in the continuous integration pipeline is to accelerate testing and help ensure new code changes do not introduce new defects or issues.

b. **Security** (code scans, container scans, license management, dependency scans)

Application security scans should be incorporated into the CI pipeline to provide rapid feedback about any software changes that introduce new vulnerabilities or issues. It is critical to have the security feedback as early as possible in the development lifecycle in order to minimize the risk of security issues delaying software delivery. The security mindset should be pervasive throughout the delivery process — from requirements to coding to testing and delivery. DevSecOps is the goal.

5. Repository to manage binary assets

The output of the continuous integration pipeline is the binary code and libraries which comprise the application. These assets must be managed and tracked through the testing, validation, and deployment of an application.

6. Continuous delivery

The continuous delivery pipeline is the automated tasks and steps required to take the application from development and package to deploy and configure in the desired environment. The CD pipeline can be a natural extension of the CI pipeline, continuing the assembly line of tasks from development into deployment and pre-production and eventually production.

7. Dynamic Test environments / infrastructure

In order to streamline development work, the software factory should support dynamic test environments (ephemeral) that can be deployed on demand to support the testing needs of individual developers and teams. Traditionally, new code changes queue up to wait for limited testing environments and resources. The factory should take advantage of containerization and cloud technology to reduce and eliminate delays that occur while waiting for test environments.

8. Incremental deployment

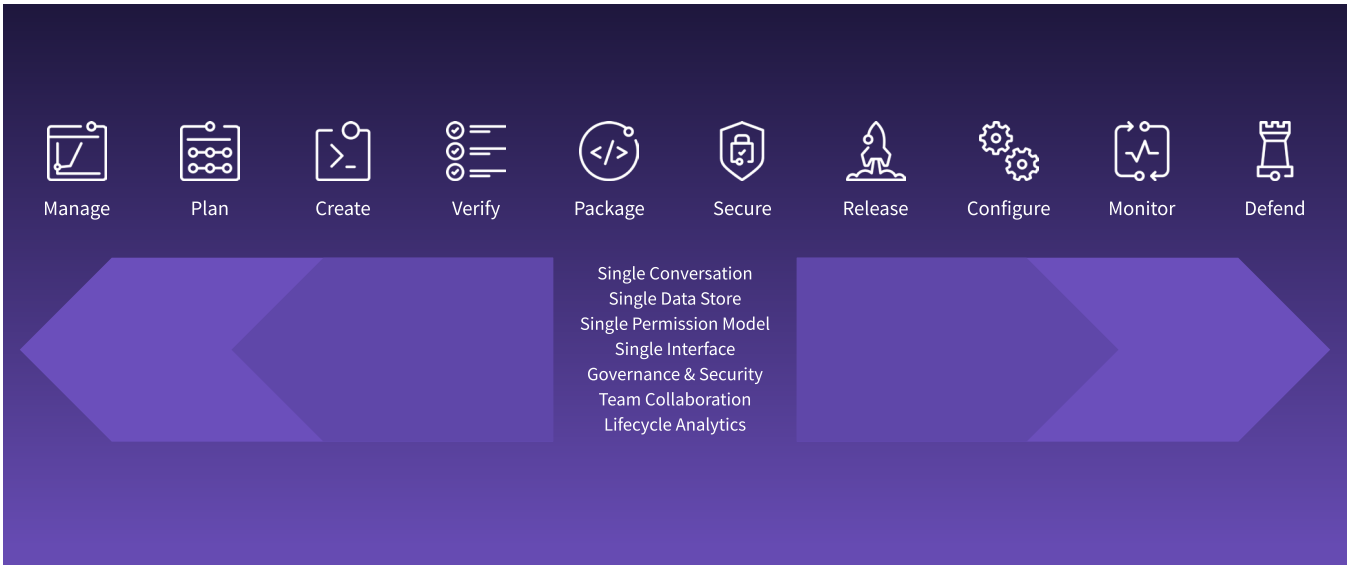
Deploying software from the factory needs to allow teams to minimize risk by supporting incremental deployments. Techniques such as canary deployments or feature flags give software development teams the flexibility to ship code quickly while actively managing and mitigating risks.

9. Application Monitoring

Feedback from the application in production is an essential part of the modern software factory. Rapid and actionable insight from application monitoring empowers product developers to detect issues, take action, and continuously improve the application.

A modern software factory encourages the collaboration, sharing, and teamwork needed to address the challenges of rapidly building and delivering applications.

GitLab is the complete software factory solution that development teams need to accelerate the delivery of mission critical capabilities. Open and extensible, GitLab is flexible enough to accommodate unique requirements from the most demanding organizations. GitLab is a single application that addresses the end-to-end requirements of the entire software delivery lifecycle.



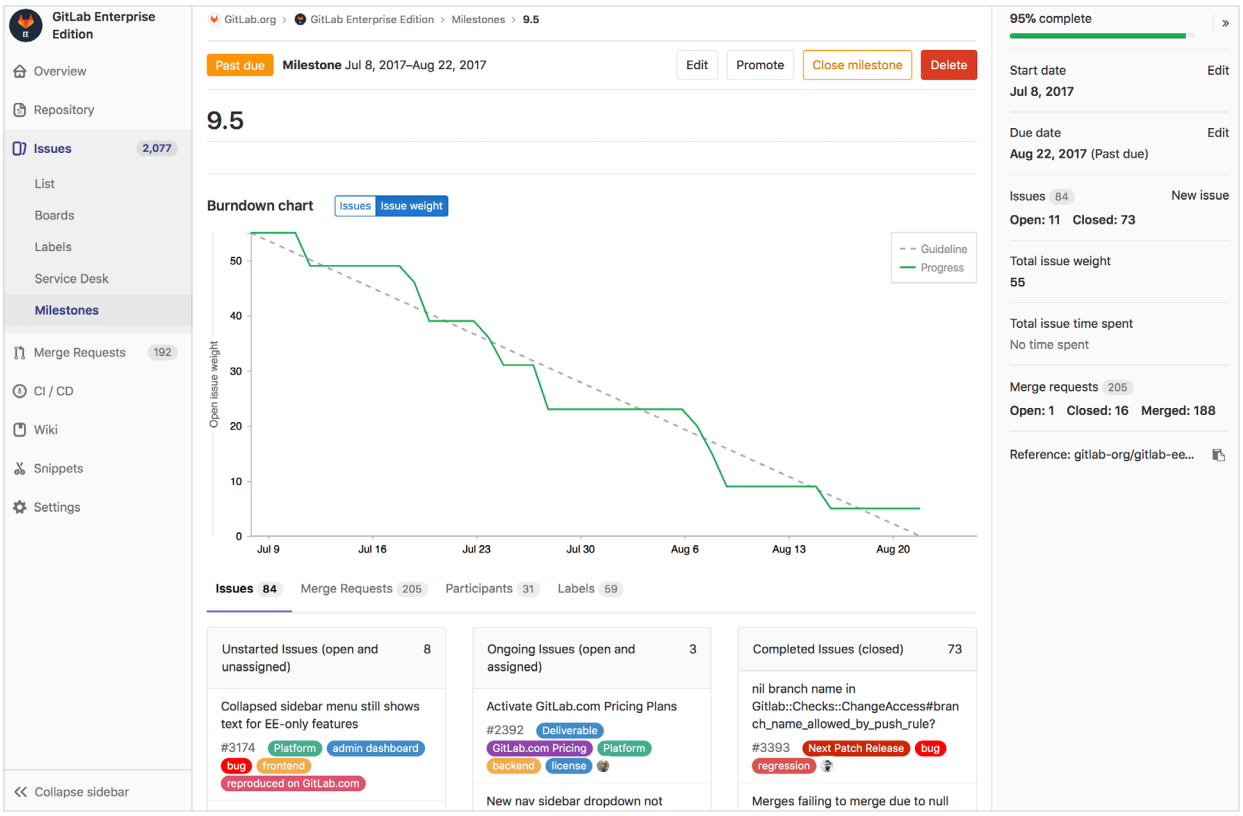
As a single application, GitLab has a unique value for delivery teams:

» A single, common user experience for the entire toolchain
» A common security and access model
» Single source of truth for reporting and managing the development work
» Simplified compliance and auditing
» A single conversation where everyone — from contracting and management to end-users and developers — participates and contributes.
» A unified governance model

Specific GitLab capabilities that create a complete software factory:

» **Agile Project Management** (epics, issues, kanban boards)
Planning for future software projects and work can be easily managed using Epics, Issues,

and Kanban Boards to prioritize and structure work. Development teams can organize and plan work into time-based sprints or based on lean principles of flow where new features are continuously developed and delivered based on business demand.



» **Distributed version control based on Git**

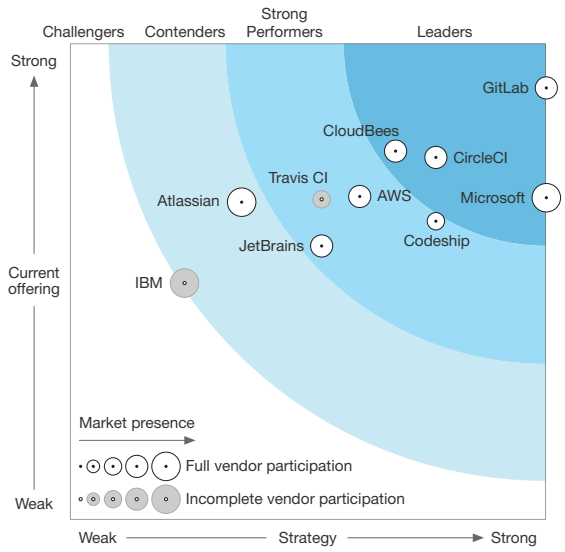
Managing multiple versions and changes to source code is efficient, managed, and visible using GitLab merge requests, which enable teams to collaborate about specific code changes, conduct code reviews, review security test results, and approve changes to the application. Merge requests can enforce policies about code reviews and approvals of code changes by appropriate individuals. Based on Git, the source control system in GitLab enables developers to efficiently work on distributed instances of the project source code and to periodically synchronize their work.

» **Industry-leading development automation:**

Continuous Integration (CI) and Continuous Delivery (CD)

The traditional workflow for a single line of code often requires manual steps to compile and build an application, conduct code quality scans, unit tests, and functional tests before deploying and configuring an updated application. In GitLab, these manual tasks are automated in GitLab CI/CD pipelines. The CI pipeline automates the tasks of building, testing, and certifying any change to the application. Automation can include tasks that run

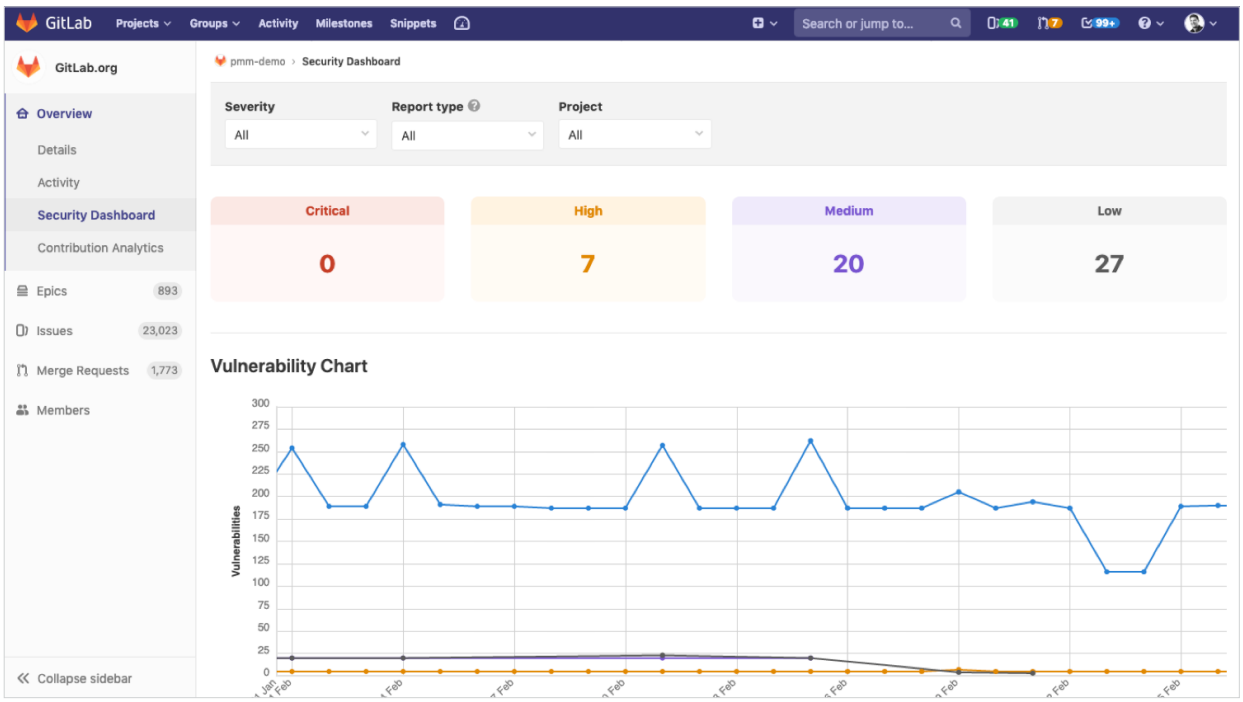
in parallel to save time, such as using pipeline automation to ensure specific compliance-related tasks are always completed for every code change. The CD pipeline automates the steps and tasks needed to provision infrastructure, install, and configure the application. In GitLab, the CD pipeline simplifies packaging the application into a container and deploying the application to an environment (OS, VM, or Kubernetes). The fundamental benefits of the CI/CD pipeline are consistent execution of steps, faster cycle time, and a reduction of manual mistakes which can have a significant impact on delivery.



Rated #1 in the Forrester CI Wave™

"GitLab supports development teams with a well-documented installation and configuration processes, an easy-to-follow UI, and a flexible per-seat pricing model that supports self service. GitLab's vision is to serve enterprise-scale, integrated software development teams that want to spend more time writing code and less time maintaining their tool chain."
— Forrester CI Wave™

» Security scans and reviews



Application security must be a first-class citizen when building modern applications. In GitLab, built-in application security scans provide rapid feedback for every code change. The security scans statically evaluate the code and dynamically evaluate the application to uncover potential vulnerabilities. Dynamic security scans evaluate the running application for potential risks. The rapid feedback gives developers the opportunity to address the security risks early in the development process, rather than waiting until the final stages prior to delivery. GitLab enables DevSecOps throughout the delivery lifecycle and incorporates container scans, license management, and application security dashboards to help teams deliver secure and reliable code.

» **Review apps**

Rapidly evaluating code changes enables developers to validate their changes and make incremental adjustments, but non-production environments for ad-hoc testing are often unavailable, forcing developers to wait. Review applications in GitLab are pre-production environments for any code change, giving developers a path to quickly run and evaluate their application, reducing the need to wait for limited resources such as test environments.

» **Canary Deploys and Feature Flags**

Releasing a large application change to end users can be a high-risk activity. Unexpected bugs or issues can have a major impact on the productivity of users and the ability of the organization to meet its mission objectives. GitLab helps to alleviate this risk to support incremental deployments, such as canary deploys where a given application change is only deployed to a subset of users. GitLab also provides feature flags, which make it possible to selectively enable or disable certain features in the application. These techniques enable teams to deliver smaller, incremental changes without introducing risk to the overall organization.

» **Application monitoring**

Complex applications consist of dozens of individual components, which are often interdependent on each other. Developers and production teams need early warnings to alert them when features are not operating as designed. GitLab includes application monitoring and tracing to empower the developers and support team to detect, diagnose, and triage problems in their application.

Other capabilities:

» **LDAP integration and CaC card support**

Identity and access management are critical concerns in any application development

project. GitLab integrates with LDAP servers and supports tokens, such as CaC cards, to manage user identity and access to the GitLab application.

» **Comprehensive Project export**
A common use case when developing an application for either classified settings or restricted environments is being able to break the development work up into unclassified components which can then be developed in lower security settings. GitLab's powerful project export feature makes it easy for teams to work in a lower security environment and then export their entire project code, discussion history, requirements, pipelines, and configuration, which can then be handed off to the team working inside the secure environment. Effectively, the complete context of the project is exported and made available to the new team.

» **High availability and disaster recovery**
Application downtime can result in an expensive loss of developer productivity. GitLab can be configured to run in a high availability mode, eliminating single points of failure. GitLab can also be configured to run in a geographically distributed mode, where multiple read-only instances are kept in sync to enable faster response times for local developers while also mitigating against disaster scenarios.

Accelerate Authority to Operate (ATO)

Government organizations often struggle to rapidly deploy new or updated applications because of the challenges obtaining their organization's governance approval known as an Authority to Operate (ATO). What development teams need is a clean and modern software factory with a fully functional assembly line that is efficient, easy to manage, and able to quickly build, test, and deliver their applications. Governance policies and tests can be automated and then included in the CI/CD build, test and deployment processes. With this approach ATO approval timelines can be significantly reduced.

Additionally, GitLab is undertaking the necessary steps to validate a hardened implementation. By passing these steps, agencies of all types can be assured of a fully secured, vulnerability-free implementation. From financial institutions in the commercial sector to government agencies such as the Department of Defense, this step will provide a high level of trust to the DevSecOps lifecycle.

Eliminate the DevOps Tool Tax

GitLab simplifies the process to design, develop, test, integrate, and ATO the components of your software factory. This lowers your tool chain 'tax' (the overhead of maintaining multiple stand alone tools) and allows organizations to focus their resources on building and deploying mission critical applications.

GitLab is trusted by hundreds of thousands of organizations as their software development and delivery solution. It is deployed in AWS, Azure, AWS GovCloud, AWS C2S, and others to support civilian, DOD, and IC agencies in their application development projects.

Modern software architectures and techniques have matured to a point where it is possible to reduce complexity and increase speed-to-value while keeping acquisition costs in check. The adoption of DevSecOps and Agile development practices enable teams to practice iterative software development and prioritize the delivery of mission critical functional requirements while streamlining communication and cultural norms. It is essential that cybersecurity and security controls are at the foundation of modernizing the software delivery process. At GitLab, we are proud to be a part of this digital transformation, helping agencies achieve speed to mission and meet the vital needs of their constituents.

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