Cross-Domain DevSecOps: Low-to-High Side Collaboration through DevSecOps

GitLab

I. Introduction

Most organizations support their Software Development Life Cycle (SDLC)—from project planning through code creation, testing, deployment, and monitoring—with an ad hoc assortment of processes and tools that have grown organically, over time. This unplanned approach may save time initially, but results in serious risks and inefficiencies as demands grow and business scales. Collaboration suffers from the complexity of the processes and tools put in place as well as the siloed teams and bottlenecks that result.

These organizations are impacted by the following challenges:

Challenges of Complex Toolchains	Impact on SDLC Efficiency
Complex processes and tools: Up to a dozen different tools to support the SDLC, each with its own user experience, data model, and integration requirements	• Multiple points of potential failure • Multiple High Assurance (HA) / Disaster Recovery (DR) requirements • Level of effort required to build and maintain integrations drains resources and innovation
Siloed teams: Role-specific teams tied to unique toolsets and Key Performance Indicators (KPIs)	• Less horizontal visibility • Reduced ability to forecast • Reduced cross-departmental cooperation • Increased duplication of work
Process fragmentation: Team and use case-specific processes for handoffs	• Inefficiencies throughout the toolchain • Increased reliance on 'tribal' knowledge • Inability to react to demands upstream or downstream in the toolchain
Last-minute validation: Functional and security testing performed late in the development cycle	• Substantial volatility in delivery schedule and cost • Developer frustration at having to determine root cause impact on all dependencies

Process bottlenecks: Manual gatekeeping between stages of the SDLC	● Inconsistent handoff quality ● Volatile velocity ● Unpredictable time-to-value

To overcome these challenges, organizations that need to successfully operate across multiple enclaves or domains simultaneously, need to simplify their toolchains first. Simplifying the tools and processes supporting their SDLC will result in the ability to create a Continuous Reauthorization policy and process as well as enabling higher talent acquisition and retention rates.

II. Use Cases

Migration from Low-to-High Side

The purpose of a network enclave is to limit internal access to a network or a portion of a network. In the US Federal arena, there are enclaves that are associated with classifications of networks ranging from Unclassified to Top Secret. For the purposes of this paper, "Low-side" is considered Unclassified and "High-side" includes any classification above Unclassified such as Confidential, Secret, and/or Top Secret. Many Federal agencies operate across several enclaves at the same time, and they need or want to develop the majority of efforts in a Low-side enclave then test, move, re-test, and deploy it in a High-side environment.

The keys to automating deployments between logically disconnected secure networks requires two major efforts:

1. Simplifying the DevSecOps Toolchain
2. Creating a Continuous Reauthorization Policy and Process

A positive by-product of these efforts is an easier and more productive talent acquisition process and higher retention rates of desirable, Highly-qualified, cleared candidates and employees.

The Simplified Toolchain:

BLUF: By simplifying the toolchain, you reduce the complexity and reduce the number of tools required/needed to export/import. The result is a more easily obtainable Authority to Operate (ATO) due to the reduction of both the complexity of the solution(s) and the required security paperwork.

The typical DevSecOps toolchain consists of 10-12 distinct tools, requiring a significant amount of labor-hours to manually integrate the tools using a combination of plug-ins and scripts, which results in establishing a configuration management baseline. When tools are updated through 3rd party plug-ins, the updates create ‘configuration drift’ and may negatively impact not only the cybersecurity posture of the tool, but the effectiveness of the toolchain as well. Managing the vast volume of tools in a modern software delivery organization’s toolchain, in accordance with continuous monitoring programs, often necessitates hiring additional skilled personnel to maintain them.

The challenge faced with deploying from and to disconnected secure networks is further compounded because all the tools and integrations must be maintained in *each* secure network. Each separate, disjointed tool not only increases the staff skillset needed to maintain the tool in each environment, it also causes higher maintenance cost because each tool must be updated in a coordinated fashion so that the CI/CD pipeline works consistently with no downtime .

Deploying actual code sets is also complicated by having to export and deliver packages for each tool in your toolchain. Gitlab simplifies the disconnected secure network deployment by providing a single application for the entire DevSecOps lifecycle. Using one application to manage the end-to-end lifecycle removes the pain of having to select, integrate, learn, and maintain a multitude of tools, in multiple networks, at multiple classification levels. A single tool enables teams to move all the data associated with a project from one domain to another as a single API call that can be automated as desired (e.g., at each commit or by using a timed pipeline).

By reducing the complexity in the toolchain supporting the SDLC, organizations can reduce the complexity of the processes of the lifecycle itself. A single source of truth provides a platform for greater visibility and automation, which, in turn, provides an opportunity to involve business stakeholders, quality gatekeepers, and even the customer in a tighter, more iterative process.

Benefits of Toolchain Simplification	**Impact on SDLC Efficiency**
Streamlined processes and tools: Cross-functional teams—from business owners to DevSecOps pros—enabled by a cross-functional tool, with a consistent UX, a common data model, and minimal integrations.	• Dramatically reduced risk profile • Simplified HA / DR requirement • Increased portability • Integration and maintenance resources freed to innovate
Cross-stage visibility: Every stage of the SDLC is visible in a common data model.	• Cross-team KPIs and alignment around value streams • Reduced team conflict due to shared priorities
Process automation: One set of processes for	• Continuous improvement through

the entire SDLC with automated handoffs between stages.	shared knowledge ● Best practice automation ● Unified documentation (automatically generated audit trails ensure compliance, traceability, and efficiency) ● Processes can be reproduced programmatically with 100% fidelity on any system
Continuous "shift-left" quality and security: Functional and security testing performed at every commit.	● Reduced risk, higher-quality code, and shortened development cycles due to lower rework and fewer downstream impacts ● More predictable delivery schedules and cost ● Substantially faster root cause analysis
Responsive processes: Consistency and automation enable greater participation and transparency.	● Reduced time to consensus from a broader range of stakeholders ● Faster iteration increases value velocity and responsiveness to shifting business needs

Continuous Reauthorization

BLUF: The ability to create a Continuous ATO process that is highly automated with visibility lies in providing consistent reviews on a scope of changes that can be easily consumed, vetted, and traced.

Currently, ATOs are based on the assumption that the system's cybersecurity posture will not change in significant ways after a system is authorized. This assumption is flawed because unchanging environments are unrealistic in modern DevSecOps practices. DevSecOps facilitates and embraces change. Organizations must reassess and reauthorize the system in a new way that can address risk and security without a major schedule slip and cost impact to program budgets.

The Risk Management Framework (RMF) provides a process that integrates security and risk management activities into the system development lifecycle. The risk-based approach to security control selection and specification considers effectiveness, efficiency, and constraints due to applicable laws, directives, Executive Orders, policies, standards, and regulations. The RMF involves all levels of an organization, from developers to top executives, and provides a new path for the ATO process through Continuous Reauthorization. Created by the National Institute of Standards and Technology (NIST), the RMF offers a guided path to integrate information security and risk management activities into the SDLC. The RMF continuous reauthorization concept perfectly aligns with core concepts and principles of DevSecOps.

Continuous Reauthorization is about changing the perspective of authorization from an event in the SDLC to a process that dynamically examines attributes that change and continually looks to validate the assessment of the systems. Systems using Continuous Reauthorization are more secure because it:

1. Reduces errors
2. Provides continuous feedback to development, operations, and security teams
3. Reduces time to deploy and resolve errors
4. Provides visibility on the security impact of every change

Stakeholders, security officers, auditors and operations gain transparency into every step of pipelines through visual inspection and an audit trail. Continuous Reauthorization also allows the implementation of governance measures and real-time compliance reporting without waiting for project managers to generate and share reports. Finally, Continuous Reauthorization removes the traditional back-and-forth between the security office and development teams.

Continuous Reauthorization eliminates error-prone human checking via a large spreadsheet of security requirements. Pipelines that automate security controls and tests also avoid bottlenecks and deliver capabilities faster by automating the tasks, reviews, tests, and approval gates that don't actually require a manual review. Continuous Reauthorization also continuously monitors the system to ensure compliance with the requirements. Automation of every phase of the DevSpecOps cycle is key to enabling Continuous Reauthorization; it provides continuous feedback, metrics, and measurement as well as transparency and traceability. Automation also supports engagement for all stakeholders, which establishes and maintains trust and mitigates risk.

Talent Acquisition/Retention

BLUF: ***Automated deployments reduce the load on high-demand employees and increase job satisfaction while enabling teams to balance the use of resources more effectively and efficiently across the wider organizational/external resource pool.***

Cleared developers are increasingly harder to find in the current market. Government agencies and companies serving the public sector are all competing for an increasingly smaller set of cleared professionals. At the same time, it has become equally difficult to get uncleared developers through the clearance process in a timely manner to deliver the rapid capabilities demanded by an ever-changing threat landscape. This shortage of cleared talent requires organizations to be able to develop unclassified code via uncleared developers to meet faster mission timelines.

Working seamlessly across multiple siloed enclaves, without losing context, enables government leaders and contractors to think differently about how they staff new and existing development programs. Recruiting efforts can now include the best and brightest, regardless of pre-existing clearances or even the ability to get clearances. Collaboration across teams increases, which improves morale and the ability to retain top talent.

III. How It Works & Why We Picked the Tools We Used

A common use case when developing an application for either classified settings or restricted environments is being able to break the development work into unclassified components which can then be developed in lower security settings. GitLab's Export feature makes it easy for teams to work in a lower security environment and then export their 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 separate environment. Team members working in that separate environment have complete context of the project with a toolset with which they're already familiar.

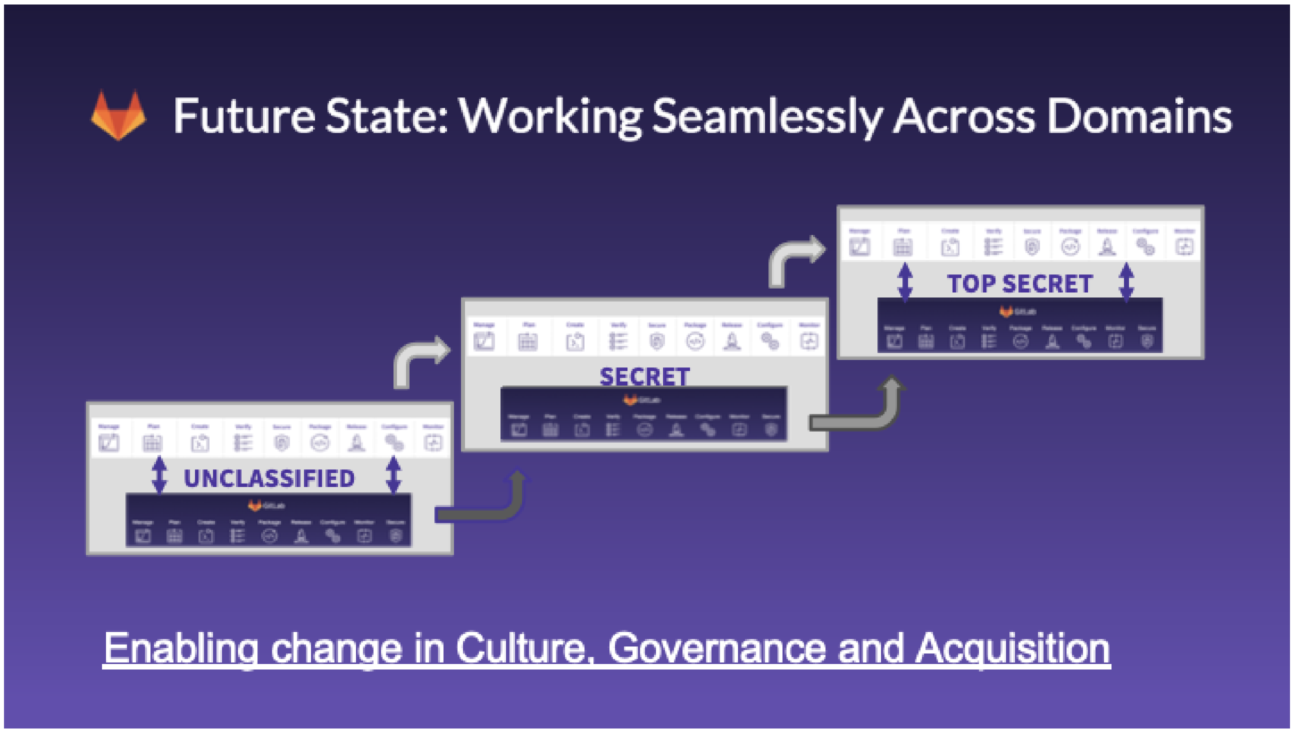
To do this at scale and speed GitLab's RESTful API makes it possible to export at the Project (single repository) and Group (multiple projects and their relationships) levels in a programmatic way that can be coordinated by the GitLab application (gitlab-ci.yml) or optionally a 3rd party tool.

Following are the steps to automate this workflow where object storage (AWS S3) will be used as the intermediary stage to hold the Export (Low-side) for retrieval by the Import (High-side). When segmenting a signal from the Low-side to trigger actions on the High-side can be set up based on many different elements including filenames & exports. This signaling often requires jobs to run asynchronously with a defined delay between the CI jobs. To do this automatically in GitLab, separate jobs are scheduled as tasks for Export (daily) and Import (daily, plus an appropriate offset). The timing of Export and Import tasks can be performed more frequently, as necessary. The size of the repo will have an impact on the timing.

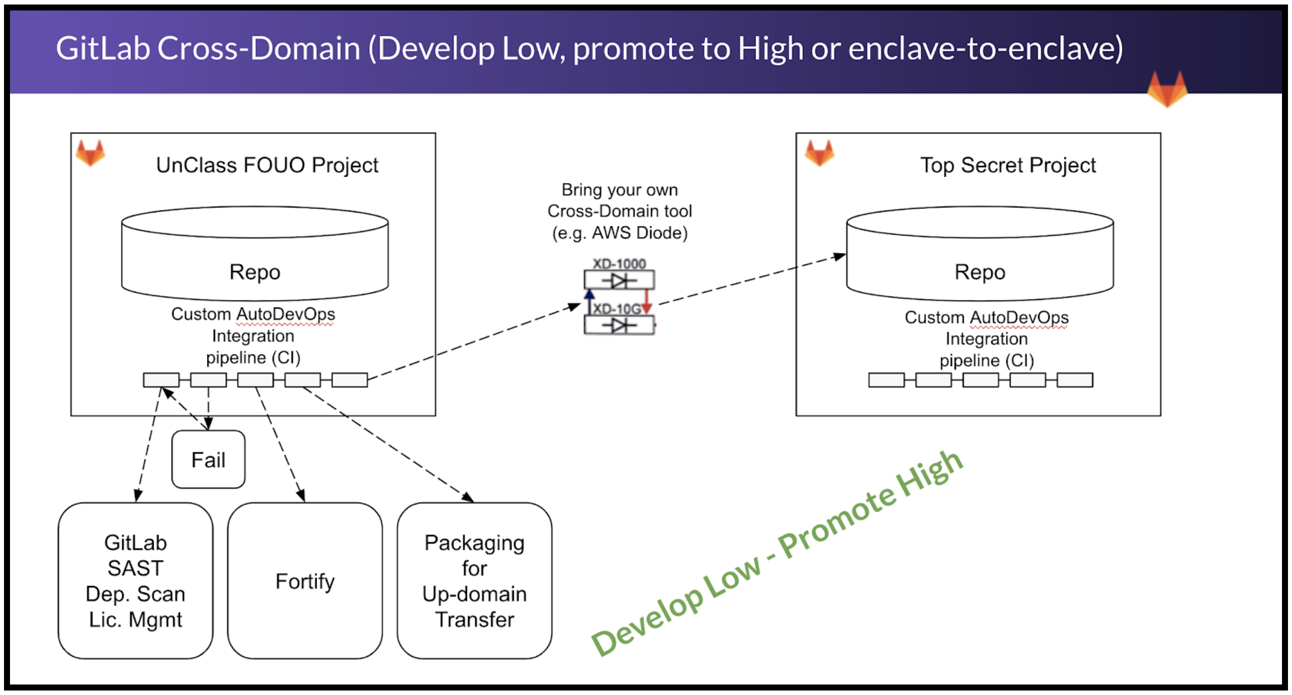
Steps for Export (Low side) to Import (High side):

1. Low-side team members collaborate on the project openly with a broad range of expertise readily available.
2. A Continuous Integration (CI) job is created within Gitlab to export the Group/Project via API calls. That exported file is then stored in an intermediary storage service such as an Amazon S3 bucket. (This can vary based upon your requirements.)
3. The Cross-Domain Solution (CDS) is now responsible for moving the exported file between the enclaves and delivering it to another S3 bucket (This can vary depending on your requirements.)

4. A CI job is created within Gitlab to import the Gitlab file. This job connects to the S3 storage, transfers the file to the Gitlab Server, and conducts the Import function via API calls.
5. If additional work needs to be done by a smaller High-side team, the repo should be forked so that work can continue (and not get overwritten by the next import). The smaller High-side team continues to work on up-to-date code in the same context and familiar toolchain as it was when it was on the Low-side.
6. Another job is created to automatically clean up the exports that are placed in the S3 buckets.
7. The CI jobs created within Gitlab are typically automatic jobs for export and import. Import and export jobs can be created to be triggered by events (code updates, merge requests approved, specific date and times, etc.) or triggered manually, where a staff member must go into the pipeline and start the CI job. Pipelines can also be set to evaluate flags and variables to allow for automatic jobs in some conditions and manual jobs in other conditions. For example, a system that is deployed on mission can be set to require manual updates where staff can determine the appropriate time to deploy or update the system.


Future State: Working Seamlessly Across Domains
UNCLASSIFIED
SECRET
TOP SECRET
Enabling change in Culture, Governance and Acquisition

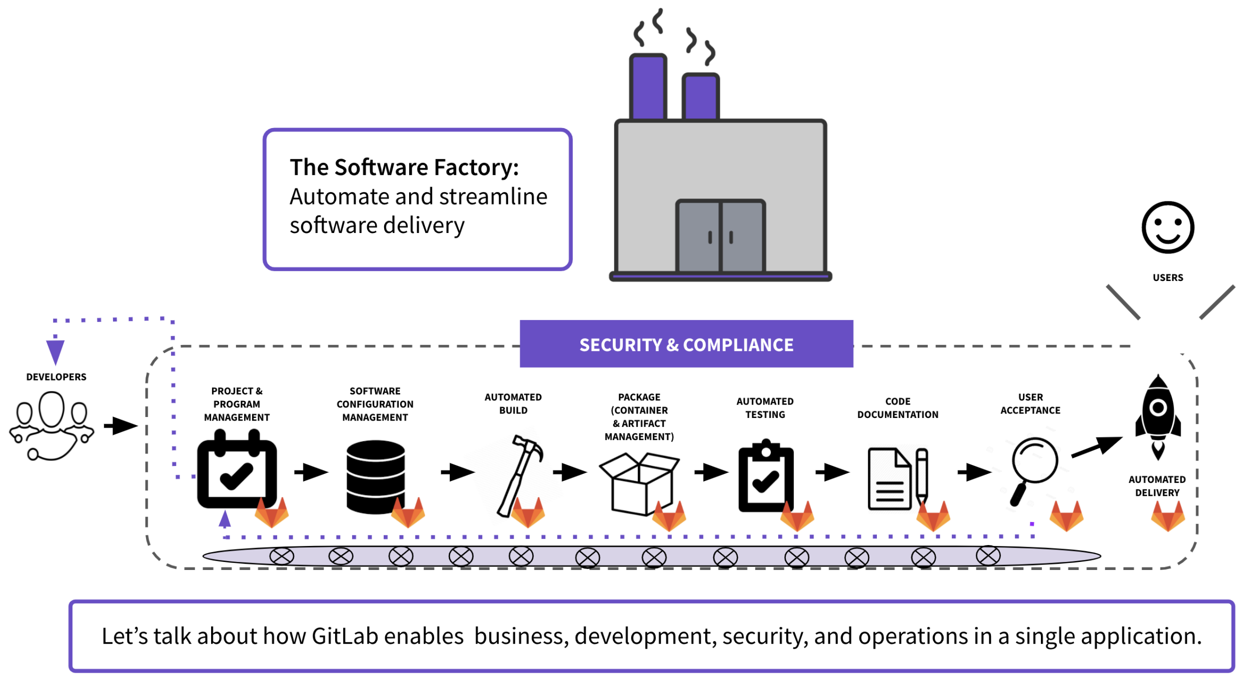
Simplifying the SDLC toolchain enables seamless export-import collaboration across domains.


GitLab Cross-Domain (Develop Low, promote to High or enclave-to-enclave)
UnClass FOUO Project
Repo
Custom AutoDevOps Integration pipeline (CI)
Fail
GitLab SAST Dep. Scan Lic. Mgmt
Fortify
Packaging for Up-domain Transfer
Bring your own Cross-Domain tool (e.g. AWS Diode)
XD-1000
XD-10G
Top Secret Project
Repo
Custom AutoDevOps Integration pipeline (CI)
Develop Low - Promote High

Enable uncleared team members to develop code on the Low-side and export to the High-Side to rapidly meet mission requirements and timelines from Day 1.

IV. How This Relates to Continuous 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 CI/CD build, test and deployment processes. With this approach, ATO approval timelines can be significantly reduced and is in line with soon-to-be-released streamlining from NIST for Automated Control-Based Assessments with Open Security Controls Assessment Language (OSCAL). This type of automation makes it possible to run automated ATO checks as frequently as every release.


The Software Factory: Automate and streamline software delivery
USERS
SECURITY & COMPLIANCE
DEVELOPERS
PROJECT & PROGRAM MANAGEMENT
SOFTWARE CONFIGURATION MANAGEMENT
AUTOMATED BUILD
PACKAGE (CONTAINER & ARTIFACT MANAGEMENT)
AUTOMATED TESTING
CODE DOCUMENTATION
USER ACCEPTANCE
AUTOMATED DELIVERY
Let's talk about how GitLab enables business, development, security, and operations in a single application.

V. Recommendations/Best Practices for Creating Low-to-High Collaboration Capability

Application segmentation is the separation of one environment from another by physical or logical constructs. Those constructs may lead to segmented environments that are remarkably similar but are built through entirely different processes (toolchains). This separation would make sense, but only if each environment were autonomous, without collaboration, and perfectly siloed.

Where each environment has the same parent organization, the question of efficiencies through reuse will eventually come up. A common use-case is the separation of application development and then deployment environments like Development, Test, and Production. Across those environments there's a need for application segmentation while allowing for reuse of organizational resources. For the most part, the application environments utilize the same pool of talent in ways that make use of segmentation with selective collaboration efforts.

The generic use-case is one where environments must be segmented and the talent working on a project can only collaborate selectively. A common toolset in each environment allows for operational familiarity and continuity between environments while maintaining application segmentation. A typical use-case is one where a large number of application resources (developers, UX/UI designers, QA, project managers, and other roles) work collaboratively in a For Official Use Only (FOUO)("Low-side") environment with a subset being eligible to work on the same application in a Top Secret ("High-side") environment. Without a consistent toolchain across environments, the context from the Low-side is lost when only the code (without artifacts, versioning, audit trails, reviews, developer comments, testing results like unit and security testing, etc.) is moved to the High-side.

GitLab's best practice recommendations are:

- Provide a consistent toolchain across all environments
- Recognize that code + collaborative details together convey context across environments
- Establish automation via RESTful API
- Enable selective feedback from the High-side to the Low-side

VI. Obtaining Buy-in from Cybersecurity Personnel

For many Federal organizations involved in DevSecOps, the desired end state includes a Cross-Domain Solution (CDS). Presently, the available solutions are limited, not well publicized, and may not be readily available or referenceable via reciprocity within your service/agency. GitLab does not recommend any specific vendor of CDS; however, the Export/Import Collaboration solution requires that the CDS hardware be able to transfer .tgz files comprised of JSON files and project bundles consisting of binary and clear text data.

There may also be some hesitation by cybersecurity personnel to authorize a system with a Cross-Domain Solution component as it adds another level of complexity beyond typical Information Systems (IS) or Platform Information Technology (PIT) systems due to crossing security enclaves or boundaries. Ultimately, system owners will need to obtain buy-in from cybersecurity personnel to implement Cross-Domain DevSecOps within their organization.

Two keys to obtain buy-in from cybersecurity personnel include:

1. Focus on the mission requirements (e.g., need to deliver faster, mission efficiency, executive direction, etc.).
2. Assessment and Authorization (A&A) Reciprocity—emphasize that CDS solutions have already been approved by other organizations and that they may be able to obtain reciprocity or a significant jumpstart to the A&A.

When adding complexity to an RMF Assessment and Authorization (A&A) package, such as the inclusion of the CDS, and cybersecurity personnel are not familiar with the technology, there is almost an immediate reaction to push back on the request to include a CDS solution. They may respond with phrases like, "I don't know of anyone that was able to get that approved" or "that may work in that *[insert service/agency]*, but we would not accept that level of risk". This is a natural reaction until they gain a better understanding of the technology, can properly assess it, and determine the security risks. Despite these reservations, the impetus to move forward with the RMF process often comes down to doing the right thing to meet the demands of the mission. Many security teams will find merit and confidence when presented with existing solutions that are working and approved. To gain additional credibility and buy-in from the cybersecurity team, it may be possible to leverage reference architectures and previously A&A approved/authorized solutions from other customers/agencies.

To stay ahead of ever-changing threats, organizations are compelled to release secure capabilities and features as fast as possible to meet the needs of their end-users. Organizations should also ask themselves, "How fast can we respond to zero-day attacks?" If the CI/CD toolchain is incredibly complex, it is inherently brittle and thus difficult to maintain as evidenced by the first use-case. This has a direct correlation to the reduction in "speed to mission" by lengthening schedules and subsequently increasing labor costs. That's all before even factoring in annual license costs for each tool in each domain or enclave. Correlating the ultimate goal of the mission with the increased capabilities to support that mission the solution provides with cybersecurity personnel will help them to overcome reservations and can lead to buy-in for the proposed CDS solution.

VII. IC Customer Success Stories

GitLab worked with a partner who directly supports a government mission program to create an internship program that allowed uncleared interns to collaborate on an unclassified instance of GitLab. The instance was used for development and collaboration for that program with the results being moved up to the customer domain (High-side) for deployment. This collaboration demonstrated speed-to-mission value within just a single summer using this framework. Because of this capability, the customer was able to continue to use those interns throughout the following year, solving the challenge of not being able to hire enough cleared people for the program. This effort applied the DevSec framework, which wraps security processes around the unclassified effort. GitLab served as the unifying technology that put everyone 'on the same page' and enabled seamless communication.

Another customer is using GitLab to enable software development and automated testing on the Low-side across multiple projects. Each project is automatically exported and sent to the High-side on a daily basis where each is subsequently imported on the High-side instance of GitLab. The full history of the project is accessible to the High-side teams so those teams can see all of the discussions, comments, code reviews, and other data related to the project. Some teams even fork the repository and continue developing on the High-side. This allows agency teams to build more applications faster as they can assign Low-side developers to tasks that allow for quicker, maximize pooled resources and lower overall organization cost. The Low-side developers' output can then be leveraged on the High-side to complete the final product while leveraging fewer classified developers.

VIII. Conclusion

Simply put, by reducing the number of tools in their SDLC toolchain, organizations can deliver on their mission faster—across multiple domains and enclaves while leveraging uncleared personnel—and more easily obtain ATO. Reducing toolchain complexity and highly automating processes enables Continuous Reauthorization and streamlines ATO approval timelines. Simplifying the toolchain also enables agencies to take advantage of Low-to-High Export/Import capabilities to attract and retain talent that can contribute from Day 1, regardless of clearance status. Simplifying the toolchain results in faster speed to mission, increased SDLC efficiencies, higher job satisfaction, and greater collaboration across the agency.

Want more information on enabling Low-to-High Side Collaboration through DevSecOps in your organization? Visit GitLab Public Sector and contact us to learn how agency customers have used this capability to move their mission forward.

IX. About Gitlab

GitLab is a complete DevSecOps platform, delivered as a single application, fundamentally changing the way Development, Security, and Ops teams collaborate. GitLab helps teams accelerate software delivery from weeks to minutes, reduce development costs, and reduce the risk of application vulnerabilities while increasing developer productivity. 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. Now, fast paced teams no longer have to integrate or synchronize multiple DevOps tools and are able to go faster by working seamlessly across the complete life cycle.

GitLab delivers complete real-time visibility of all projects and relevant activities across the entire DevSecOps life cycle. For the first time, teams can see everything that matters. Changes, status, cycle times, security and operational health are instantly available from a trusted single source of data. Information is shown where it matters most, e.g. production impact is shown together with the code changes that caused it. And developers see all relevant security and ops information for any change. 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 give everyone the data-driven confidence to act with new certainty. With GitLab, DevSecOps 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.

GitLab drives radically faster cycle times by helping DevSecOps teams achieve higher levels of efficiency across all stages of the life cycle making it possible for Product, Development, QA, Security, and Operations teams to work at the same time, instead of waiting for handoffs. Teams can collaborate and review changes together before pushing to production. GitLab eliminates the need to manually configure and integrate multiple tools for each project. GitLab makes it easy for teams to get started, they can start with GitLab using one or two use cases they need to improve, and then begin their evolution to a single end-to-end experience for all of DevSecOps.

Only GitLab delivers DevSecOps teams powerful new governance capabilities embedded across the expanded lifecycle to automate security, code quality and vulnerability management. With GitLab, tighter governance and control never slow down DevOps speed.

GitLab leads the next advancement of DevSecOps. Built on Open Source, GitLab delivers new innovations and features on the same day of every month by leveraging contributions from a passionate, global community of 4800+ developers and millions of users. Over 100,000 of the world's most demanding organizations trust GitLab to deliver great software at new speeds.

about.gitlab.com/solutions/public-sector